

# NAR-5100 Series Communication Appliance

## User's Manual

Revision: 010



### **Portwell Inc.**

3F, No. 92, Sec. 1, Nei-Hu Rd., Taipei 114, Taiwan, R.O.C.  
Headquarter: +886-2-2799-2020 FAX: +886-2-2799-1010  
<http://www.portwell.com.tw>  
Email: [info@mail.portwell.com.tw](mailto:info@mail.portwell.com.tw)



# Table of Contents

<b>Chapter 1</b>	<b>Introduction .....</b>	<b>2</b>
1.1	<i>About This Manual .....</i>	<i>2</i>
1.2	<i>Manual Organization .....</i>	<i>2</i>
1.3	<i>Technical Support Information .....</i>	<i>2</i>
<b>Chapter 2</b>	<b>Getting Started .....</b>	<b>4</b>
2.1	<i>Included Hardware .....</i>	<i>4</i>
2.2	<i>Before You Begin .....</i>	<i>4</i>
2.3	<i>The Chassis .....</i>	<i>5</i>
2.4	<i>Open the Chassis.....</i>	<i>5</i>
2.5	<i>Remove and Install DIMM.....</i>	<i>6</i>
2.6	<i>Remove and Install Compact Flash Card.....</i>	<i>6</i>
2.7	<i>Remove and Install Battery .....</i>	<i>7</i>
2.8	<i>Install HDD .....</i>	<i>7</i>
2.11	<i>Ear Mount Kit Installation .....</i>	<i>8</i>
2.12	<i>Product Specifications.....</i>	<i>9</i>
2.13	<i>Hardware Configuration Setting .....</i>	<i>9</i>
2.14	<i>Install a Different Processor .....</i>	<i>12</i>
2.15	<i>Use a Client Computer .....</i>	<i>12</i>
2.16	<i>BIOS Setup Information .....</i>	<i>14</i>
2.17	<i>Reset to Default Information .....</i>	<i>20</i>
2.18	<i>WDT Information .....</i>	<i>29</i>
2.19	<i>GPIO Information .....</i>	<i>32</i>
2-20	<i>About EZIO-100 .....</i>	<i>40</i>
<b>Chapter 3</b>	<b>Operation Guide.....</b>	<b>50</b>
3.1	<i>Brief Guide of PPAP-3714L .....</i>	<i>50</i>
3.2	<i>System Architecture .....</i>	<i>51</i>

# Chapter 1 Introduction

## 1.1 About This Manual

This manual contains all required information for setting up and using the NAR-5100 series.

NAR-5100 is an Intel® Pentium® 4-based system with the Cavium NITROX IPsec/SSL processor onboard. The embedded Cavium's NITROX Lite CN1010X processor features cost benefit and flexibility by supporting IPsec, SSL processing .

NAR-5100 provides the essential platform for delivering optimal performance and functionality in the value communications appliance market segment. This manual should familiarize you with NAR-5100 series operations and functions. NAR-5100 series has up to ten on-board Ethernet ports to serve communication appliances like Firewall, requiring Ethernet ports to connect external network (internet), demilitarized zone and internal network.

Feature of NAR-5100 series includes:

- ◆ One On-board Cavium Nitrox CN1010X Security Processor
- ◆ Versatile networking and I/O capabilities: 10 Ethernet ports
- ◆ Dual 64bit/133MHz Gigabits Ethernet (Intel 82546 EB)
- ◆ Up to 4GB /266/333/400MHz DDR on 4 x 184-pin DIMM sockets, with ECC/ Unregistered .
- ◆ One on-board IDE channel to support one removable 2.5 " HDD
- ◆ User-friendly LCD control panel

## 1.2 Manual Organization

This manual describes how to configure your NAR-5100 series system to meet various operating requirements. It is divided into three chapters, with each chapter addressing the basic concept and operation of this system.

- Chapter 1: Introduction. This section describes how this document is organized. It includes brief guidelines and overview to help find necessary information.
- Chapter 2: Hardware Configuration Setting and Installation. This chapter demonstrated the hardware assembly procedure, including detailed information. It shows the definitions and locations of Jumpers and Connectors that can be used to configure the system. Descriptions on how to properly mount the CPU and main memory are also included to help perform a safe installation. This chapter will provide detailed instruction on how to set up NAR-5100 series.
- Chapter 3: Operation Information. This section provides illustrations and information on the system architecture and how to optimize its performance.

Any updates to this manual, technical clarification and answers to frequently asked questions would be posted on the web site: <http://isc.portwell.com.tw>

## 1.3 Technical Support Information

Users may find helpful tips or related information on Portwell's web site: <http://www.portwell.com.tw>. A direct contact to Portwell's technical person is also available. For further support, users may

also contact Portwell's headquarter in Taipei or local distributors.

***Taipei Office Phone Number: +886-2-27992020***

## Chapter 2      Getting Started

This section describes how the hardware installation and system settings should be done.

### *2.1 Included Hardware*

The following hardware is included in package:

- ◆ PPAP-3714L Communication Appliance System Board
- ◆ One null serial port cable

### *2.2 Before You Begin*

To prevent damage to any system board, it is important to handle it with care. The following measures are generally sufficient to protect your equipment from static electricity discharge:

When handling the board, use a grounded wrist strap designed for static discharge elimination and touch a grounded metal object before removing the board from the antistatic bag. Handle the board by its edges only; do not touch its components, peripheral chips, memory modules or gold contacts.

When handling processor chips or memory modules, avoid touching their pins or gold edge fingers. Restore the communications appliance system board and peripherals back into the antistatic bag when they are not in use or not installed in the chassis.

Some circuitry on the system board can continue operating even though the power is switched off. Under no circumstances should the Lithium battery cell used to power the real-time clock be allowed to be shorted. The battery cell may heat up under these conditions and present a burn hazard.

#### **WARNING!**

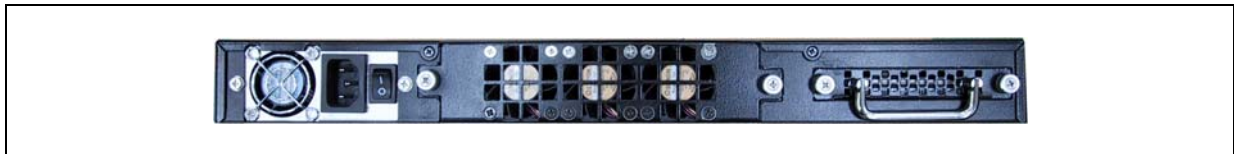
1. **"CAUTION: DANGER OF EXPLOSION IF BATTERY IS INCORRECTLY REPLACED. REPLACE ONLY WITH SAME OR EQUIVALENT TYPE RECOMMENDED BY THE MANUFACTURER. DISCARD USED BATTERIES ACCORDING TO THE MANUFACTURER'S INSTRUCTIONS"**
2. **This guide is for technically qualified personnel who have experience installing and configuring system boards. Disconnect the system board power supply from its power source before you connect/disconnect cables or install/remove any system board components. Failure to do this can result in personnel injury or equipment damage.**
3. **Avoid short-circuiting the lithium battery; this can cause it to superheat and cause burns if touched.**
4. **Do not operate the processor without a thermal solution. Damage to the processor can occur in seconds.**
5. **Do not block air vents. Minimum 1/2-inch clearance required.**

## 2.3 The Chassis

The system is integrated in a customized 1U chassis (**Fig. 2-1, Fig. 2-2**). On the front panel you will find a 4-push-button LCD module (EZIO), totally ten LAN ports and a COM port.



**Fig. 2-1** Front view of the chassis



**Fig. 2-2** Rear view of the chassis

## 2.4 Open the Chassis

1. Loosen the four screws of the chassis, two on each side and the rest two on the back, to remove the top lead . (**Fig. 2-3**).



**Fig. 2-3** Take off screws

2. The top lead (**Fig. 2-4**) can be removed from the base stand (**Fig. 2-5**).



**Fig. 2-4** The top lead

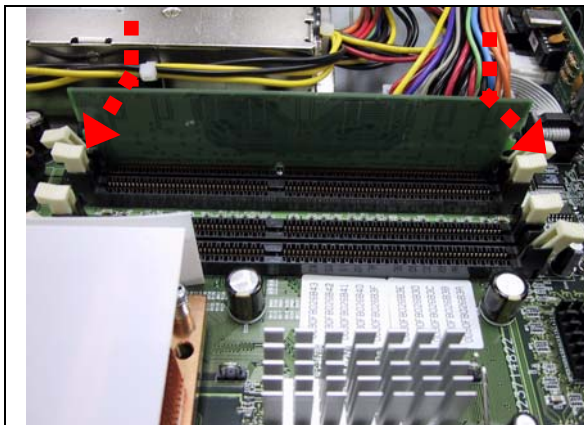


**Fig. 2-5** The base stand

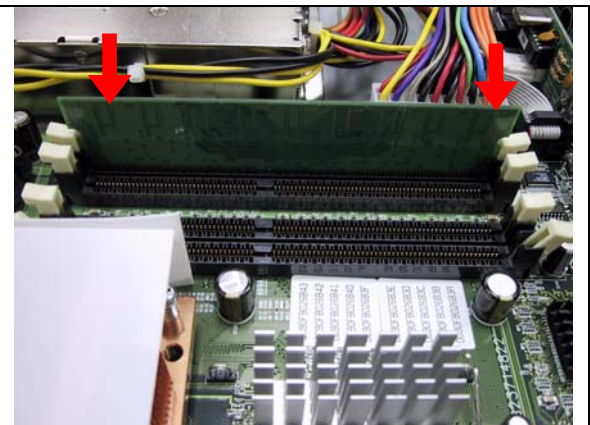
## 2.5 Remove and Install DIMM

Follow these steps to upgrade RAM module:

1. Install the system memory by pulling the socket's arm and pressing it into the slot gently. (**Fig. 2-6, 2-7**)



**Fig. 2-6** Eject a DIMM module



**Fig. 2-7** Install DIMM

## 2.6 Remove and Install Compact Flash Card

1. Insert the Compact Flash Card (**Fig. 2-8**) into the CF interface (**Fig. 2-9**).



**Fig. 2-8** Compact Flash Card



**Fig. 2-9** Insert Compact Flash Card into the CF



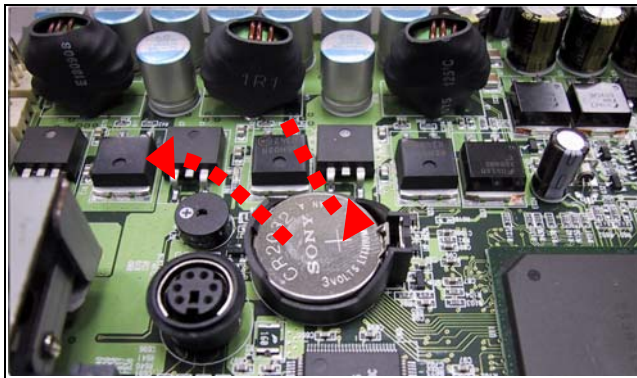
2. The completed installation of Compact Flash Card is shown as **Fig. 2-10**.



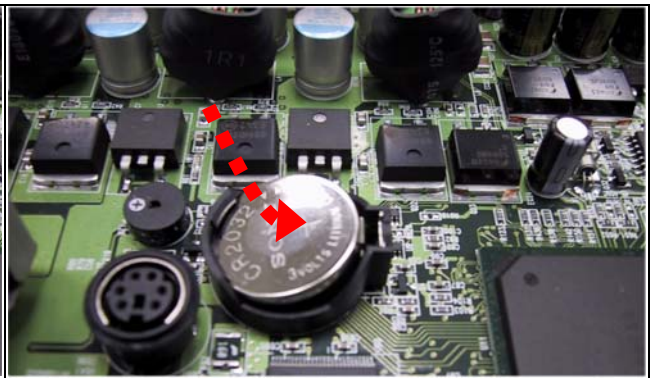
**Fig. 2-10** Completion of Compact Flash Card connection

## 2.7 Remove and Install Battery

1. Press the metal clip back to eject the button battery (**Fig. 2-11**).
2. Replace it with a new one by pressing the battery with fingertip to restore the battery (**Fig. 2-12**).



**Fig. 2-11** Eject the battery



**Fig. 2-12** Restore the battery

## 2.8 Install HDD

The system has an internal drive bay for one 2.5" hard disk drive. If the HDD is not pre-installed, you can install it by yourself. Follow the steps below to install the HDD:

1. Install HDD connector with HDD. (**Fig. 2-13a, 2-13b**).





**Fig. 2-13a** A 2.5" HDD and the HDD bracket

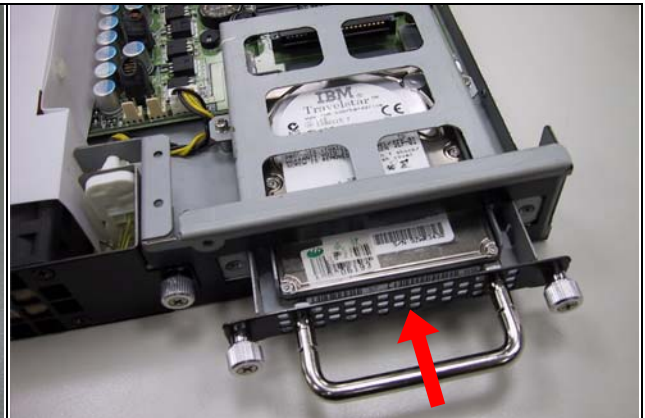


**Fig. 2-13b** Install HDD Connector to HDD

2. Fasten the six screws to lock HDD and bracket together (**Fig. 2-14**).
3. Connect HDD to NAR-5100 (**Fig. 2-15**).



**Fig. 2-14** Fix HDD to the bracket

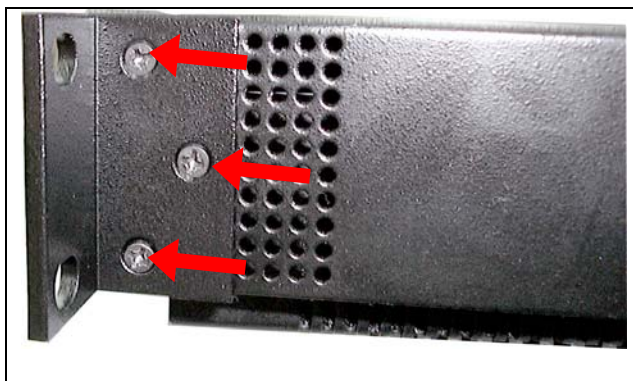


**Fig. 2-15** Connect IDE to PPAP-3714L

## 2.11 Ear Mount Kit Installation

The NAR-5100 shipped with 2 ear mount kits. The following is the installation instruction of these ear mounts:

1. Take out the L shape ear mount kits. One ear mount fits on one side of the chassis,
2. Placing the side with three holes against the chassis and the side with two holes face outward. (**Fig. 2-26**)
3. Fast three screws on each side (**Fig. 2-26**)



**Fig.2-26** Fasten the screws to the side

## 2.12 Product Specifications

<b>Model:</b>	<b>NAR-5100</b>
<b>Main Processor :</b>	<ul style="list-style-type: none"><li>• Intel® socket 478 Pentium® 4 processors ( 533MHz or 800MHz system bus)</li></ul>
<b>BIOS:</b>	<ul style="list-style-type: none"><li>• Award system BIOS with 512KB flash ROM to support DMI, PnP, APM function</li></ul>
<b>Main Memory:</b>	<ul style="list-style-type: none"><li>• Totally four 333/400 MHz unregistered DDR on 184-pin DIMM socket supports up to 4GB of 2.5V DIMM, with ECC.</li></ul>
<b>L2 Cache Memory:</b>	<ul style="list-style-type: none"><li>• 256KB/512KB PBSRAM built in (Celeron/Pentium 4) CPU module</li></ul>
<b>Chipset:</b>	<ul style="list-style-type: none"><li>• North Bridge: Intel 875P</li><li>• South Bridge: Intel 6300ESB</li></ul>
<b>PCI IDE Interface:</b>	<ul style="list-style-type: none"><li>• One 40/ IDE-1 interface for DMA/33/66/100 IDE hard disk</li></ul>
<b>Serial Ports:</b>	<ul style="list-style-type: none"><li>• Support two high-speed 16550C compatible UARTs with 16-byte T/R FIFOs</li><li>• (Optional) Support LCD/Key pad module (Portwell proprietary)</li></ul>
<b>USB Interface:</b>	<ul style="list-style-type: none"><li>• Support two USB2.0 ports for high speed I/O peripheral devices</li></ul>
<b>Auxiliary I/O Interfaces:</b>	<ul style="list-style-type: none"><li>• System reset switch, power okay LED and HDD LED interface</li></ul>
<b>Power Input:</b>	<ul style="list-style-type: none"><li>• Support one AC input jack (power requirement: 110V ~ 220V auto switch)</li></ul>
<b>PCI Slot:</b>	<ul style="list-style-type: none"><li>• One PCI slot for add-on PCI card</li></ul>
<b>On-board Ethernet:</b>	<ul style="list-style-type: none"><li>• One Intel® 82546EB 100/1000 Gigabit Fast Ethernet controllers with RJ-45 interface for NAR-5100</li><li>• One Intel® 82551QM 10/100 Fast Ethernet controllers</li><li>• Seven Intel® 82551ER 10/100 Fast Ethernet with RJ-45 interface for NAR-5100</li></ul>
<b>Hardware Monitor:</b>	Support on-board hardware monitor for <ul style="list-style-type: none"><li>• Totally four for CPU and System fan</li><li>• System voltages: Vcore, 3.3V, +5V and +12V</li></ul>
<b>Power Good:</b>	<ul style="list-style-type: none"><li>• On-board power good interval: 100ms ~ 500ms</li></ul>
<b>Environmental Requirements:</b>	<ul style="list-style-type: none"><li>• Operating Temperature: 5°C ~ 40°C</li><li>• Storage Temperature: 5°C ~ 70°C</li><li>• Relative Humidity: 5% ~ 95%, non-condensing</li></ul>
<b>Dimension:</b>	<ul style="list-style-type: none"><li>• 429(W) x 330(D) x 44(H)mm</li></ul>

## 2.13 Hardware Configuration Setting

This section gives the definitions and shows the positions of jumpers, headers and connectors. All of the configuration jumpers on PPAP-3714L are in the proper position. The default settings

set by factory are marked with a star ( ★ ).



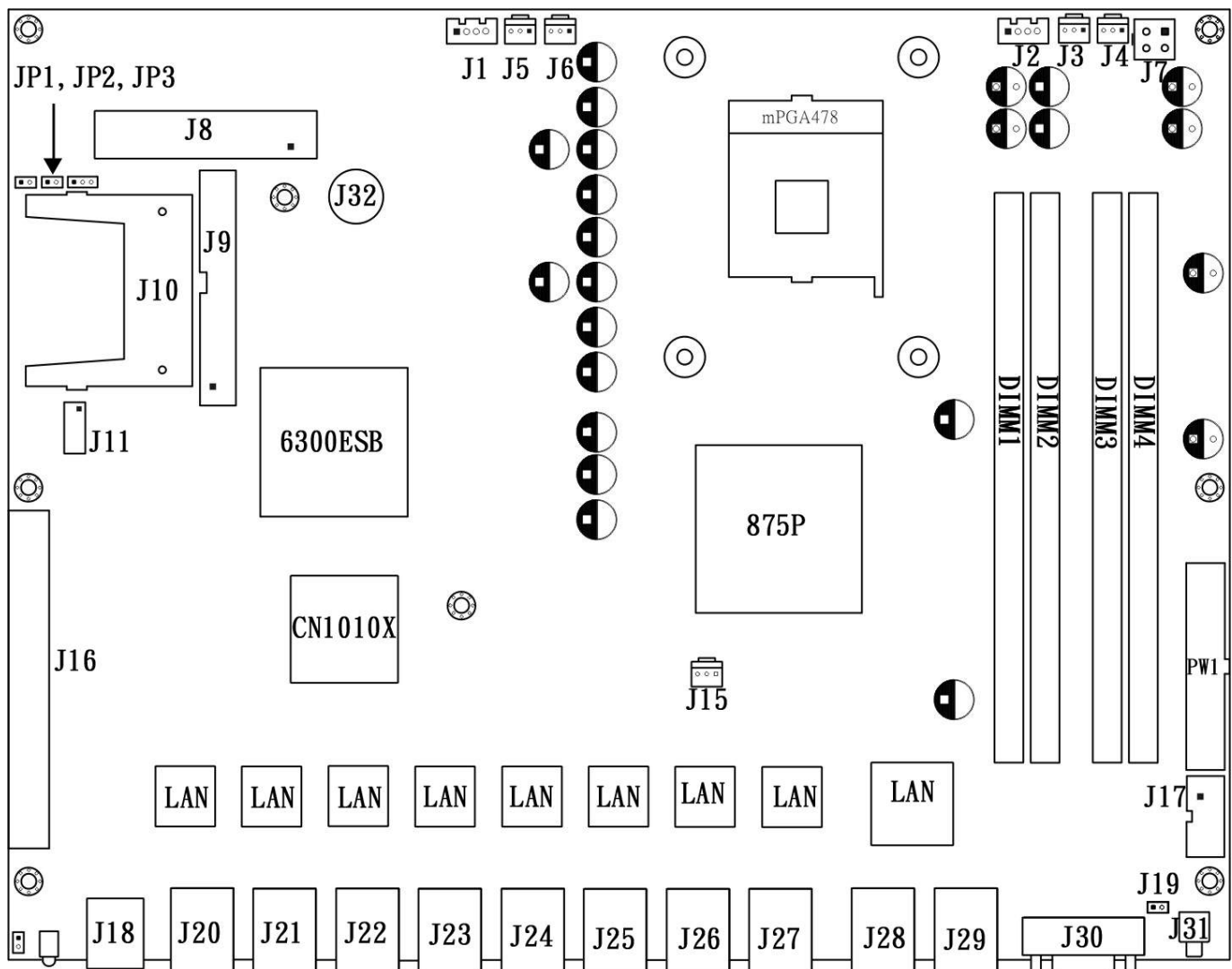
## Jumpers

In general, jumpers on PPAP-3714L system board are used to select options for certain features. Some of the jumpers are configurable for system enhancement. The others are for testing purpose only and should not be altered. To select any option, cover the jumper cap over (Short) or remove (NC) it from the jumper pins according to the following instructions. Here NC stands for “Not Connected”.

**Jumper Table**

Jumper	Function	Default Setting
<b>JP1</b>	<u>Watchdog function :</u> Short : Reset Open : NM1/SM1	Watchdog reset
<b>JP2</b>	<u>CF Setup:</u> Short : Master Open : Slave	Master
<b>JP3</b>	<u>CMOS Clear:</u> 1-2: CMOS normal 2-3: CMOS clear	CMOS normal

**Figure 2-27 PPAP-3714L Jumper Table**



### **Connector**

Connector	Function
J1	SYSTEM FAN SENSOR
J2	SYSTEM FAN
J3	CPU/SYSTEM FAN#1
J4	CPU/SYSTEM FAN#2
J5	CPU/SYSTEM FAN#3
J6	CPU/SYSTEM FAN#4
J8	IDE CNANNEL0 (44pin)
J9	IDE CHANNEL1 (40pin)
J10	IDE CHANNEL1 CF CARD
J12	FORNT PANEL FAN
J13	PS2 KEYBOARD/MOUSE
J15	SYSTEM FAN#4
J16	PCI
J17	COM2
J18	USB
J19	Reset to default
J20	LAN#10
J21	LAN#9
J22	LAN#8
J23	LAN#7
J24	LAN#6
J25	LAN#5
J26	LAN#4
J27	LAN#3
J28	LAN#2
J29	LAN#1
J30	COM1
J31	Reset bottom
J32	PS2 KEYBOARD/MOUSE
PW1	ATX POWER CONNECTOR

## 2.14 Install a Different Processor



### **Install CPU**

1. Lift the handling lever of CPU socket outwards and upwards to the other end.
2. Align the processor pins with holes on the socket. Make sure that the notched corner or dot mark (pin 1) of the CPU corresponds to the socket's bevel end. Then press the CPU gently until it fits into place. If this operation is not easy or smooth, don't do it forcibly. You need to check and rebuild the CPU pin uniformly.
3. Push down the lever to lock processor chip into the socket.
4. Follow the installation guide of cooling fan or heat sink to mount it on CPU surface and lock it on the socket 478.
5. Be sure to follow particular CPU speed and voltage type to adjust the jumper settings properly for all boards.



### **Remove CPU**

1. Unlock the air funnel and heat sink first.
2. Lift the lever of CPU socket outwards and upwards to the other end.
3. Carefully lift up the existing CPU to remove it from the socket.
4. Follow the steps of CPU installation to change to another one or place handling bar to close the opened socket.



### **Configure Processor Speed**

The system was designed to self-detect its CPU speed. So it does not require any system adjustment.

## 2.15 Use a Client Computer



### **Connection Using Hyper Terminal**

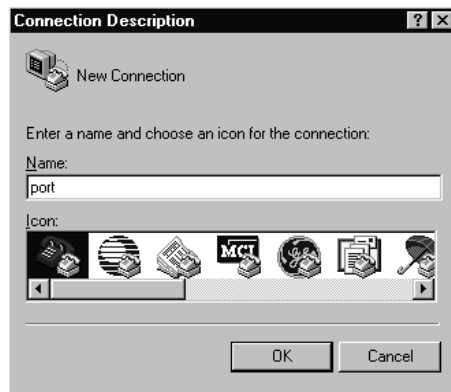
If users use a NAR-5100 series, which has no mouse/keyboard and VGA output connected to it, the console may be used to communicate with NAR-5100 series.

To access NAR-5100 series via the console, Hyper Terminal is one of many choices. Follow the steps below for the setup:

**Note:** Terminal software may need to update for correct console output.

1. Execute HyperTerminal under C:\Program Files\Accessories\HyperTerminal

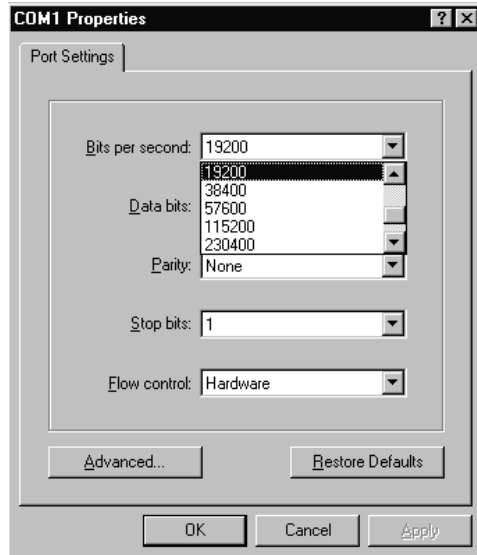
2. Enter a name to create new dial



3. For the connection settings, make it Direct to Com1.



4. Please make the port settings to Baud rate 19200, Parity None, Data bits 8, Stop bits 1

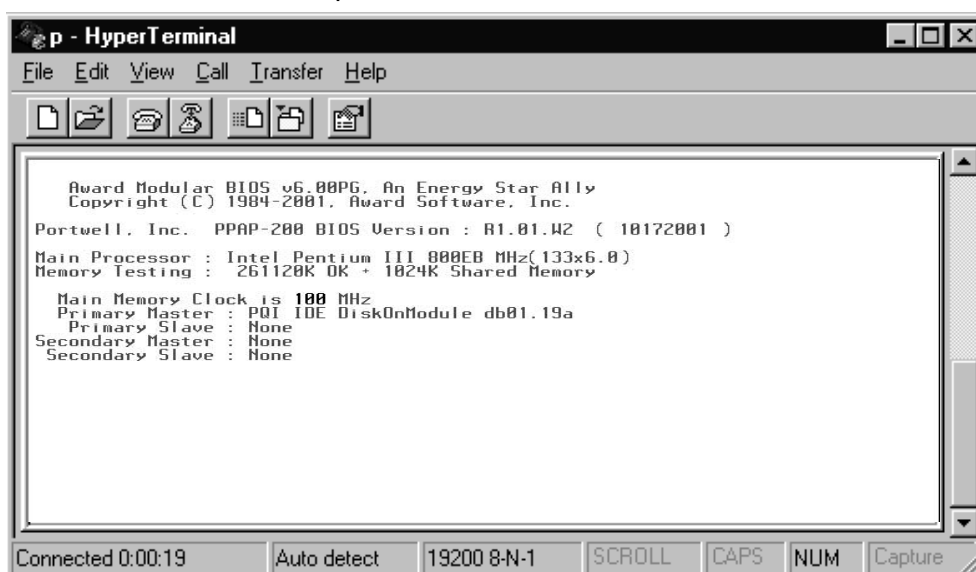


5. Turn on the power of NAR-5100 series, after following screen was shown:





6. You can then see the boot up information of NAR-5100 series.



7. When message “Hit <DEL> if you want to run Setup” appear during POST, after turning on or rebooting the computer, press **<Tab>** key **immediately** to enter BIOS setup program.

This is the end of this section. If the terminal did not port correctly, please check the previous steps.

## 2.16 BIOS Setup Information

NAR-5100 series is equipped with the Award BIOS within Flash ROM. The BIOS has a built-in setup program that allows users to modify the basic system configuration easily. This type of information is stored in CMOS RAM so that it still retains during power-off periods. When system is turned on, NAR-5100 series communicates with peripheral devices and checks its hardware resources against the configuration information stored in the CMOS memory. Whenever an error is detected, or the CMOS parameters need to be initially defined, the diagnostic program will prompt the user to enter the Setup program. Some errors are significant enough to abort the start-up.



## **Entering Setup**

When message “Hit <DEL> if you want to run Setup” appear during POST, after turning on or rebooting the computer, press <Del> key **immediately** to enter BIOS setup program.

To enter Setup but fail to respond before the message disappears, please restart the system either by first turning it off and followed by turning it on (COLD START) or simply press the "RESET" button. “WARM START” (press <Ctrl>, <Alt>, and <Delete> keys simultaneously) will do as well.

When no setting is stored in BIOS or the setting is missing, a message “Press <F1> to run Setup” will appear. Then press <F1> to run Setup or resume HIFLEX BIOS Setup. User can use the keyboard to choose among options or modify the system parameters to match the options with your system. The table shown on next page will navigate through all of keystroke functions in BIOS Setup.

### ***Keys to navigate within Setup menu***

Key	Function
Up (↑)	Move to the previous item
Down (↓)	Move to the next item
Left (←)	Move to the item on the left (menu bar)
Right (→)	Move to the item on the right (menu bar)
Enter	Enter the item you desired
PgUp	Increase the numeric value or make changes
PgDn	Decrease the numeric value or make changes
+	Increase the numeric value or make changes
—	Decrease the numeric value or make changes
Esc	<b>Main Menu:</b> Quit and not save changes into CMOS <b>Status Page Setup Menu and Option Page Setup Menu:</b> Exit current page and return to Main Menu
F1	General help on SETUP navigation keys
F5	Load previous values from CMOS
F6	Load the fail-safe defaults from BIOS default table
F7	Load the optimized defaults
F10	Save all the CMOS changes and exit



## **Main Menu**

Within NAR-5100series Award BIOS CMOS Setup utility, user should start with the Main Menu. The Main Menu allows to select from eleven setup functions and two exit choices. Use arrow keys to switch among items and press <Enter> to accept or bring up the sub-menu.

### ***Phoenix – Award BIOS CMOS Setup Utility***

**CMOS Setup Utility**

Standard CMOS Features Advanced BIOS Features Advanced Chipset Features Integrated Peripherals Power Management Setup PnP/PCI Configurations PC Health Status	Frequency /Voltage Control Load Fail/Safe Defaults Load Optimized Defaults Set Supervisor Password Set User Password Save & Exit Setup Exit Without Saving
ESC: Quit F10: Save & Exit Setup	↑ ↓ ← →: Select Item (Shift) F2: Change Color
Time, Date, Hard Disk Type ...	

**NOTE:** It is strongly recommended to reload the optimized default setting if CMOS is lost or BIOS is updated.



### Standard CMOS Setup Menu

This setup page includes all the items within standard compatible BIOS. Use the arrow keys to highlight the item and then use the <PgUp>/<PgDn> or <+>/<-> keys to select the value or number you want in each item and press <Enter> to certify it.

Follow command keys in CMOS Setup table to change Date, Time, Drive type and Boot Sector Virus Protection Status.

### Screen Shot: Phoenix – Award BIOS CMOS Setup Utility

Standard CMOS Setup Utility	
Date: Wed, Jan 17 2004 Time: 16:51:13	
IDE Primary Master [None] IDE Primary Slave [None] IDE Secondary Master [None] IDE Secondary Slave [None] Video: EGA/VGA Halt On: All, but Keyboard	
Base Memory: 640K Extended Memory: 512MB Total Memory: 512MB	
ESC: Quit F1: Help PU/PD/+/-: Modify	↑ ↓ ← →: Select Item (Shift) F2: Change Color

### Menu Selections

Item	Options	Description
<b>Date</b>	mm:dd:yy	Set the system date. Note that the 'Day' automatically changes when you set the date
<b>Time</b>	hh:mm:ss	Set the system time
<b>Video</b>	EGA/VGA CGA 40CGA	Select the default video device

	80MONO	
<b>Halt On</b>	All Errors No Errors All, but Keyboard All, but Diskette All, but Disk/Key	Select the situation in which you want the BIOS to stop the POST process and notify you
<b>Base Memory</b>	N/A	Display the amount of conventional memory detected during boot up
<b>Extended Memory</b>	N/A	Display the amount of extended memory detected during boot-up
<b>Total Memory</b>	N/A	Display the total memory available in the system



## **BIOS Features Setup**

This section allows you to configure your system for basic operation. You are able to select the system's default speed, boot-up sequence, keyboard operation, shadowing and security.

### ***Screen Shot: Phoenix – Award BIOS CMOS Setup Utility***

<b>Advanced BIOS Features</b>		
CPU Feature	[Press Enter]	Console Redirection: Enable Agent connect via: NULL Agent wait time (min.): 1 Agent after boot: Disabled
Hard Disk Boot Priority	[Press Enter]	
Virus Warning	[Disable]	
CPU L1 & L2 Cache	[Press Enter]	
Hyper-Threading Technology	[Enabled]	
Quick Power On Self Test:	[Enabled]	ESC: Quit ↑ ↓ ← →: Select Item F1: Help (Shift) F2: Color F5: Old Values F6: Load BIOS Default F7: Load Setup Default PU/PD/+/-: Modify
First Boot Device	[USB-Floppy]	
Second Boot Device	[CD-ROM]	
Third Boot Device	[Hard-Disk]	
Boot Other Device	[Enable]	
Typematic Rate Setting	Disabled	
Typematic Rate (Chars/Sec)	6	
Typematic Delay (Msec)	250	
Security Option	[Setup]	
OS Select for DRAM > 64MB	[Non-OS2]	
Small Logo(EPA) Show	[Disable]	



## **Console Redirection**

Set the UNIX Console redirect to the terminal from COM1.

The choice: Enabled/Disabled.



## **Baud Rate**

Set the RS-232 baud rate speed.

The choice: 9600, 19200, 38400, 57600 and 115200.



### **Cache Setup**

CPU L1 & L2 Cache ( Enabled or Disabled CPU L1 and L2 cache )

<b>Enabled</b>	Enable cache
<b>Disabled</b>	Disable cache

CPU L2 Cache ECC Checking ( Enabled or Disabled CPU L2 cache ECC checking )

<b>Enabled</b>	Enable checking
<b>Disabled</b>	Disable checking



### **Virus Warning**

Allows you to choose the VIRUS warning feature for IDE Hard Disk boot sector protection.

<b>Enabled</b>	Enable VIRUS warning
<b>Disabled</b>	Disabled VIRUS warning



### **Hyper-Threading Technology**

For WindowsXP and Linux 2.4.x ( OS optimized for Hyper Threading Technology )

<b>Enabled</b>	Enable Hyper Threading Technology
<b>Disabled</b>	Disabled Hyper Threading Technology



### **Quick Power On Self Test**

This category speeds up Power On Self Test (POST) after you power up the computer. If it is set to Enable, BIOS will shorten or skip some check items during POST.

<b>Enabled</b>	Enable quick POST
<b>Disabled</b>	Normal POST



### **Boot Up NumLock Status**

Select power on state for NumLock.

The choice: Enabled/Disabled.



### **Typematic Rate Setting**

Keystrokes repeat at a rate determined by the keyboard controller. When enabled, the typematic rate and typematic delay can be selected.

The choice: Enabled/Disabled.



### **Typematic Rate (Chars/Sec)**

Set the how many number of times a second to repeat a keystroke when you hold the key down.

The choice: 6, 8, 10, 12, 15, 20, 24 and 30.



### **Typematic Delay (Msec)**

Set the delay time after the key is held down before it begins to repeat the keystroke.

The choice: 250, 500, 750 and 1000.



### **Security Option**

Select whether the password is required every time the system boots or only when entering setup.

<b>System</b>	The system will not boot and access to Setup will be denied if the correct password is not entered at the prompt.
<b>Setup</b>	The system will boot and access to Setup will be denied if the correct password is not entered at the prompt.



### **OS Select for DRAM > 64MB**

Select the operating system that is running with more than 64MB of RAM on the system.

The choice: Non-OS2, OS2.



## 2.17 Reset to Default Information

```
; For PPAP-3714, RESET to Default testing
; By Frank Hsu , 11/08/2004
; Reset to default status can be read from ESB6300_GPI6
; IF BIOS does not init GPIO_USE_SEL6 does not set to "1" ,
; then the initial level on GPI6 signal may be 0.8~0.98V.
; This Signal level is no good for logical high or low.
; " Therefore ,this program will init GPIO_USE_SEL6 to "1" .
;
; After Power On reset, GPI6 = low ( 0 )
; If Reset to Default (RST2DF) Button pressed ( Triggered )
; ,then GPI6 will be latch to high ( 1 ).
;
; RST2DF register can be cleared by ESB6300_GPO19.
; Write a pulse timing ( High1_low_high2 ) to clear RST2DF to 0.
; High1 : output GPO19 high , and keep 30 us.
; Low : output GPO19 low , and keep 30 us.
; High2 : output GPO19 high again , and keep high always.
;
;
; Programming Guide :
; PG_Step1 : Enable ACPI IO port assignment and get PMBASE.
;
; First : GPI_ROUT bit[25,24] P [0,0] : Let GPI6 not evoke SCI.
; Write GPI_Rout bit[13,12] to [0,0] for no effect on GPI6
; ( B0:D31:F0:Offset_B8h-Bit[25,24]P[0,0] , no SCI event evoked)
;
; Second: Enabe ACPI IO port by setting ACPI_CNTL bit4
; B0:D31:F0:Offset_44h_bit4P1
; Third : Get PMBASE ( ACPI I/O port BAR )
; PMBASE=:B0:D31:F0:Offset[40..43h]
; Let Bit0 = 0.( PCI_BAR bit0 returns 1 for a IO_BAR )
;
;
; PG_Step2 : Enable GPIO IO function and get GPIOBASE.
;
; How to program GPIO19 ( Output only , i.e. GPO19 )
; -----
; Get GPIOBASE =: B0:D31:F0:Offset[58..5Bh] ;(and let bit0 = 0 )
; GPIO_CNTL =: B0:D31:F0:Offset_5Ch_bit4P1 ;Enable GPIO
; GPIO_USE_SEL19[(GPIOBASE+0)_bit19]P1
; ; to define GPIO19 as GPIO function
; GP_BLINK19[(GPIOBASE+18h)_bit19]P0
; ; to let GPIO19 function normally ( No_Blink )
; GPIO19
; GP_LVL (=:(GPIOBASE + 0Ch))_bit19P[0/1]; Write value 0/1
; -----
; How to read GPI6
; =====
; GPIO_USE_SEL6[(GPIOBASE+0)_bit6]P1
; ; to define GPI6 as GPIO function
; GPI6 status MUST NOT be inverted First.
; GPI_INV (=GPIOBASE+2Ch)-bit6P0. ( GPI6 not inverted )
; Get GPI6 status from GPE0_STS (=PMBASE+28h)-bit22
; 0 = low , 1= high level
; =====
.MODEL small
.386
.STACK 200h
.data
PROMPT1 DB'PORTWELL PPAP-3714,3714RSTD.exe, V1.00 11-08-2004,All rights reserved.$'
PROMPT_1 DB ' For PPAP-3714 Reset-to-Default test .',13,10,'$'
PROMPT_2_CR_LF db 0Dh, 0Ah,0Dh, 0Ah, '$'
PROMPT_Str1 db ' Reset-To-Default status latched by a F/F. ',0dh,0ah,'$'
```

```

PROMP_Str2 db ' This status bit = 0 ---> Normal. ',0dh,0ah,'$'
PROMP_Str3 db ' This status bit = 1 ---> RST2DF button has been pressed.',0dh,0ah,'$'
PROMP_Str4 db ' This status bit can be read by ESB6300_GPI6, ',0dh,0ah,'$'
PROMP_Str5 db ' and can be cleared by an ESB300_GPO19 High1-Low-High2 pulse.',0dh,0ah,'$'
PROMP_Str6 db ' ',0dh,0ah,'$'
PROMP_Str7 db ' High1 = 30us High level ',0dh,0ah,'$'
PROMP_Str8 db ' Low = 30us Low level ',0dh,0ah,'$'
PROMP_Str9 db ' High2 = High level again and no level change from now on.',0dh,0ah,'$'
PROMP_StrA db ' ',0dh,0ah,'$'
PROMP_rst2df db 0dh,0ah,' Press the Reset-to-Default button and then release it for the test NOW!$'
PROMP_anykey db 0dh,0ah,' Ready ? If yes , then Press any key to start test ..... '$'
PROMP_err1 db 0dh,0ah,' ***** "Reset-to-Default F/F Initialization" Failed. *****',0dh,0ah,'$'
PROMP_err1_1 db ' ( This may be a H/W error or Reset-to-Default button has ever been
pressed ! )',0dh,0ah,'$'
PROMP_err2 db 0dh,0ah,' ***** "Reset-to-Default event latched by F/F " Failed. *****',0dh,0ah,'$'
PROMP_err3 db 0dh,0ah,' ***** "Clear Reset-to-Default F/F status " Failed. *****',0dh,0ah,'$'
PROMP_TEST_OK db ' <<..... PPAP-3714 RESET-TO-DEFAULT test OK .....>>',0dh,0ah,'$'
PROMP_TEST_fail db ' <<***** PPAP-3714 RESET-TO-DEFAULT test FAIL *****>>',0dh,0ah,'$'
PROMP_Qkey db 0dh,0ah,'Press "Q" key to stop test and return to DOS; or other key to go on next test.$'
debugtesting db 0dh,0ah,' <<<<< This is for debugging only >>>>> ',0dh,0ah,'$'

```

```

GP_INV_OFFSET      db 2Ch ; The offset value from GPIOBASE
GPE0_STS_OFFSET    db 28h ; The offset value from PMBASE
GP_LVL_OFFSET      db 0Ch ; The offset value from GPIOBASE
GP_BLINK_OFFSET     db 18h ; Must let GPIO19 to be Blink_disabled for
                    ; normal GPIO function
GPIO_USE_SEL_OFFSET db 00h ; GPIO_USE_SEL_Bit19P1 as GPIO function
PMBASE              dw ? ;
GPIOBASE            dw ? ;

```

.code

programstart:

```

mov ax,@data
mov ds,ax
lea dx,PROMP_2_CR_LF
mov ah,09h
int 21h
lea dx,PROMP1
mov ah,09h
int 21h
lea dx,PROMP_2_CR_LF
mov ah,09h
int 21h
lea dx,PROMP1_1
mov ah,09h
int 21h

```

```

lea dx,PROMP_2_CR_LF
mov ah,09h
int 21h

```

```

lea dx,PROMP_Str1
mov ah,09h
int 21h
lea dx,PROMP_Str2
mov ah,09h
int 21h
lea dx,PROMP_Str3
mov ah,09h
int 21h
lea dx,PROMP_Str4
mov ah,09h
int 21h
lea dx,PROMP_Str5
mov ah,09h
int 21h
lea dx,PROMP_Str6

```

```

mov ah,09h
int 21h
lea dx,PROMP_Str7
mov ah,09h
int 21h
lea dx,PROMP_Str8
mov ah,09h
int 21h
lea dx,PROMP_Str9
mov ah,09h
int 21h
lea dx,PROMP_StrA
mov ah,09h
int 21h

```

```

mov edx,00000000h ; Error flag in EDX_BIT[16..18], 0=ok, 1=failed

```

```

; PG_Step1 : Enable ACPI IO port assignment and get PMBASE, then save to
;
; First : GPI_ROUT bit[25,24] P [0,0] : Let GPI6 not evoke SCI.
;       Write GPI_Rout bit[25,24] to [0,0] for no effect on GPI6
;       ( B0:D31:F0:Offset_B8h-Bit[25,24]P[0,0] , no SCI event evoked)
;
; Second: Enabe ACPI IO port by setting ACPI_CNTL bit4
;       B0:D31:F0:Offset_44h_bit4P1
; Third : Get PMBASE ( ACPI I/O port BAR )
;       PMBASE=B0:D31:F0:Offset[40..43h]
;       Let Bit0 = 0.( PCI_BAR bit0 returns 1 for a IO_BAR )
;
;

```

```

; ----- 1_start
; Get PMBASE
; Let GPI6 GPI_ROUT to [0,0] , i.e. not evoke SCI in S0.

```

```

mov dx,0CF8h ; PCI Config Read
mov eax,8000F8B8h ; B0:D31:F0:Offset_B8h
out dx,eax
mov dx,0CFCh
in eax,dx

```

```

ror eax,10h ;
and ah,0FCh ; bit[25,24] set to [0,0] to let GPI6 not
rol eax,10h ; evoke SCI event

```

```

out dx,eax

```

```

mov dx,0CF8h ; PCI Config Read
mov eax,8000F844h ; B0:D31:F0:Offset_44h
out dx,eax
mov dx,0CFCh
in eax,dx

```

```

or al,10h ; bit 4 set to 1 to enable PMBASE
out dx,eax ;

```

```

mov dx,0CF8h ; Get PMBASE
mov eax,8000F840h ; B0:D31:F0:Offset_40h
out dx,eax
mov dx,0CFCh
in eax,dx

```

```

and al,0feh ; bit0 cleared to 0.

```

```

    mov PMBASE,ax
; ----- 1_end

; PG_Step2 : Enable GPIO IO function and get GPIOBASE,
;
;   How to program GPO19
;   -----
;   Get GPIOBASE =: B0:D31:F0:Offset[58..5Bh] ;(and let bit0 = 0 )
;   GPIO_CNTL =: B0:D31:F0:Offset_5Ch_bit4P1 ;Enable ICH4 GPIO
;
;   GPIO_USE_SEL19[(GPIOBASE+0)_bit19]P1
;   ; to define GPIO19 as GPIO function
;
;   GP_BLINK19[(GPIOBASE+18h)_bit19]P0
;   ; to let GPIO19 function normally ( No_Blink )
;
;   GPO19
;   GP_LVL  (=:(GPIOBASE + 0Ch))_bit19P[0/1]; Write value 0/1
;   -----
;
;   How to read GPI6
;   =====
;   GPIO_USE_SEL6 bit must be set to "1" to make sure GPI6 as
;   a GPIO function First .
;
;   GPI6 status must NOT be inverted then.
;   GPI_INV (=GPIOBASE+2Ch)-bit6P0. ( GPI6 not inverted )
;
;   Get GPI6 status from GPE0_STS (=PMBASE+28h)-bit22
;   0 = low , 1= high level
;   =====
;
; ===== 2_start
; Get GPIOBASE Base Address ,

    mov dx,0CF8h
    mov eax,8000F85Ch ; B0:D31:F0:Offset_5Ch
    out dx,eax
    mov dx,0CFCh
    in  eax,dx

    or  al,10h      ; 5Ch_Bit4P1 to Enable GPIO
    out dx,eax

    mov dx,0CF8h    ; Get GPIOBASE
    mov eax,8000F858h ; B0:D31:F0:Offset_58h
    out dx,eax
    mov dx,0CFCh
    in  eax,dx

    and al,0feh     ; bit 0 cleared to 0.
    mov GPIOBASE,ax ;

    xor ax,ax
    mov dx,GPIOBASE
    mov al,GPIO_USE_SEL_OFFSET
    add dx,ax
    in  al,dx
    or  al,40h      ; GPIO_USE_SEL6P1
    out dx,al

    xor ax,ax
    mov dx,GPIOBASE

```

```

    mov al,GPIO_USE_SEL_OFFSET
    add dx,ax
    add dx,2      ; Point to Bit[16..23]
    in  al,dx
    or  al,08h    ; Bit19P1 GPIO_USE_SEL19P1
    out dx,al     ; GPIO19 MUST be set as GPIO function

    xor ax,ax
    mov dx,GPIOBASE
    mov al,GP_BLINK_OFFSET
    add dx,ax
    add dx,2      ; Point to Bit[16..23]
    in  al,dx
    and al,0F7h   ; Bit19P0
    out dx,al     ; GPIO19 MUST be set as NO_Blinking ( Write 0 )

; ===== 2_end

; Testing way :
; --- t1
; Read GPI6 first , GPI6=0 ? if yes,pass ; if no, failed
;
; --- t2
; RST2DF button pressed and released , read GPI6 ,GPI6 = 1 ? if yes, pass ; if no, failed
;
; --- t3
; Clear RST2DF status to 0 ,read GPI6 ,GPI6 = 0 ? if yes, pass ; if no, failed
;----- t_start

; make sure GPO19 = 1 start ( RST2DF F/F no cleared by GPO19 )
xor bx,bx

    mov bl,GP_LVL_OFFSET    ; Write GPO19 1
    mov dx,GPIOBASE        ;
    add dx,bx              ;
    add dx,02h             ; point to GPIO[16..23] register
    in  al,dx              ; read first

    call IODELAY ; io delay

    or  al,08h             ; bit3 ---> GPO19
    out dx,al              ; output GPO19 1
; make sure GPO19 = 1 end
;===== MUST DO =====Start
; GP_INV bit6 MUST Program 0 for GPI6 state not inverted. start

    xor bx,bx

    mov bl,GP_INV_OFFSET    ; Not invert GPI6 status
    mov dx,GPIOBASE        ;
    add dx,bx              ;
    in  al,dx              ; read first

    call IODELAY ; io delay

    and al,0BFh            ; mask bit6 and write 0
    out dx,al

; GP_INV bit6 MUST Program 0 for GPI6 state not inverted. end
;===== MUST DO =====End

; ---- t1 start
; GPI6 , read its status , initialization will be 0 .
;

```

```

;      How to read GPI6
;      -----
;      Get GPI6 status from GPE0_STS (=PMBASE+28h)-bit22
;      0 = low , 1= high level
;      -----

call READ_GPI6_TO_AL

and al,40h ; mask bit6
cmp al,00h

    je next_test1 ; okay , go on test

; no , error message display

    lea dx,prompt_err1
    mov ah,09h
    int 21h
    lea dx,prompt_err1_1
    mov ah,09h
    int 21h

    ror edx,10h ; error falg EDX_Bit16 , 1 --> Error happened
    or  dl,01h
    rol  edx,10h

    call KB_Wait

; ---- t1 end

    next_test1 :

; ---- t2 start
    lea dx,prompt_rst2df
    mov ah,09h
    int 21h

    lea dx,prompt_anykey
    mov ah,09h
    int 21h

    xor al,al ; halt for ready? Any key pressed to go on.
WAIT_KB_0:
    mov ah,1
    int 21h

    cmp al,0
    je WAIT_KB_0

    lea dx,PROMP_2_CR_LF
    mov ah,09h
    int 21h

; test RST2DF button pressed

    call READ_GPI6_TO_AL

and al,40h ; mask bit6
cmp al,40h

    je next_test2 ; okay , go on test

; no , error message display

```



```

    lea dx,prompt_err2
    mov ah,09h
    int 21h

    ror edx,10h ; error flag EDX_Bit17 , 1 --> Error happened
    or dl,02h
    rol edx,10h

    call KB_Wait

; ---- t2 end

    next_test2 :

; ---- t3 start ,Clear RST2DF F/F
; GPO19 write 1,0,1
;
; ===== Write GPO19 1-0-1 start

    xor bx,bx

    mov bl,GP_LVL_OFFSET ; Write GPO19 1
    mov dx,GPIOBASE
    add dx,bx ;
    add dx,02h ; point to GPIO[16..23] register
    in al,dx ; read first

    call IODELAY ; io delay

    or al,08h ;
    out dx,al ; output GPO19 1 first

    call FIXDELAY ; 30 us delay

    in al,dx ; output GPO19 0 then

    call IODELAY

    and al,0F7h
    out dx,al

    call FIXDELAY ; 30 us delay

    in al,dx ; output GPO19 high finally

    call IODELAY

    or al,08h
    out dx,al
; ===== Write GPO19 1-0-1 end

    call READ_GPI6_TO_AL ; check RST2DF F/F

    and al,40h ; mask Bit6
    cmp al,00h
    je test_end ; okay , then end

; jz test_end ; okay , then end

; no , error message display

    lea dx,prompt_err3
    mov ah,09h

```

```

        int 21h

        ror edx,10h ; error falg EDX_Bit18 , 1 --> Error happened
        or  dl,04h
        rol  edx,10h

; ---- t3 end

test_end :

        ror edx,10h ; check error flag
        cmp dl,00h
        je  test_ok

test_fail :

        lea dx,PROMP_2_CR_LF
        mov ah,09h
        int 21h

        lea dx,prompt_TEST_fail
        mov ah,09h
        int 21h

        jmp return_to_dos

test_ok :
        lea dx,prompt_TEST_OK
        mov ah,09h
        int 21h

return_to_dos :
        mov ah,4ch ; Return to DOS
        int 21h

;----- t_end
; ===== 2_end

IODELAY PROC  near
        push ax
        push dx

        mov dx,0edh
        in  al,dx
        jmp $+2
        mov dx,0edh
        in  al,dx

        pop dx
        pop ax
        ret
IODELAY ENDP

KB_wait PROC  near

        push ax
        push bx
        push cx
        push dx

        lea dx,PROMP_Qkey
        mov ah,9 ; Display "Q" key prompt
        int 21h

```

```

        xor al,al
WAIT_KB:
        mov ah,1
        int 21h

        cmp al,0
        je WAIT_KB

        cmp al,51h ; "Q" pressed ?
        je test_fail
        cmp al,71h ; "q" pressed ?
        jne call_return
        jmp test_fail ;

call_return :

        lea dx,PROMP_2_CR_LF
        mov ah,09h
        int 21h

        pop dx
        pop cx
        pop bx
        pop ax

        ret

KB_wait ENDP

READ_GPI6_TO_AL PROC    near
    push bx
    push dx

    xor bx,bx

    mov dx,PMBASE
    mov bl,GPE0_STS_OFFSET
    add bl,02 ; Point to Bit22 for GPI6
    add dx,bx
    in al,dx

    call IODELAY ; io delay

; MUST to do write 1 to clear GPE0_STS_bit22 to 0 FIRST due to the
; access ( 0/1 ) . This register is R/WC , and will be set
; at any time when GPI signal is high.
;

    and al,40h ; mask bit6
    or al,40h ; WC
    out dx,al ; Write bit4 to 0 first.

    call IODELAY ; io delay

    in al,dx ; read GPI6 again

    pop dx
    pop bx
    ret
READ_GPI6_TO_AL ENDP

;-----;

```

```

;          FIXED_DELAY          ;
;-----;
;   Input : (CX) count of 15 microseconds to wait      ;
;   STACK PRESENT                                     ;
;   Output: NONE                                       ;
;   CX=2 , 15us x 2 = 30 us                           ;
;
; This routine is called to wait for 15 microseconds * count in ;
; (CX), then return. Gives a programmed software delay.      ;
;-----;
FIXDELAY PROC  near
    push cx
    push dx
    push ax
    pushf

    mov cx,02h

    mov dx,61h
    in  al,dx      ;
    jmp $+2
    jmp $+2
    and al,00010000b ;
    mov ah,al      ;
fixed_delay_1:
    in  al,dx      ;
    jmp $+2
    jmp $+2
    and al,00010000b ;
    cmp al,ah      ;
    jz  short fixed_delay_1 ;
    mov ah,al      ;
    loop short fixed_delay_1 ;

    popf
    pop ax          ;
    pop dx
    pop cx
    ret
FIXDELAY ENDP

END programstart

```

## 2.18 WDT Information

```

/*This is a test program for 32-bit protected mode memory access. */
/*The compiler id DJGPP, please refer http://www.delorie.com/djgpp */
/*This program is to test Intel 6300ESB ICH internal WDT. */
#include <dpmi.h>
#include <sys/types.h>
#include <sys/movedata.h>
#include <go32.h>
#include <pc.h>
#include <keys.h>
#include <unistd.h>
#include <dos.h>
#include <conio.h>

/* Constant define */
#define VenderID 0x8086
#define DeviceID 0x25ab /* Intel 6300ESB ICH WDT D2:F4 */
#define Mem_Size 4096

```

```

#define KEY_1 0x31
#define KEY_2 0x32
#define KEY_3 0x33

#define Preload_Value_1_offset 0x00
#define Preload_Value_2_offset 0x04
#define Int_Sts_offset 0x08
#define Reload_Reg_offset 0x0c

/* Varriety define */
unsigned long Base_Addr; /* to save memory base address of four 878A */
int Sel; /* LDT selector */
unsigned char time_out;
__dpmi_regs regs;

/* subprogram */
void Search_PCI_WDT(void)
{
    int i;

    /* Search PCI Deviceo */
    printf("\n\nPortwell Inc. 6300ESB ICH WDT TEST PROGRAM V1.00\n\n");
    printf("\nSearch 6300ESB ICH WDT Device");
    /* Get PCI Device's Addr info */
    regs.x.ax = 0xb102; /* find PCI device */
    regs.x.dx = VenderID; /* Vender ID */
    regs.x.cx = DeviceID; /* Device ID */
    regs.x.si = i; /* Device index */
    __dpmi_int(0x1a, &regs);
    if (regs.h.ah == 0)
        printf("\nWDT device is find!", i);
    else
    {
        printf("\nDevice%d not find!!", i);
        _exit(1);
    }
    /* get Base0 data */
    regs.x.ax = 0xb10a; /* get PCI Cfg. Dword */
    regs.x.di = 0x0010; /* get CR.0x10 */
    __dpmi_int(0x1a, &regs);
    regs.h.cl = 0;
    Base_Addr = regs.d.ecx;
    if (regs.h.ah == 0)
        printf("\nPCI Base Address: %x", Base_Addr);
    else
    {
        printf("\nRead PCI Cfg. Reg. fail!!");
        _exit(1);
    }
}

void prepare_access_above_1mb(void)
{
    int j;
    __dpmi_meminfo MemInfo;

    /* Process for memory access above 1MB */
    /* Map the physical device address to linear memory */
    MemInfo.address = Base_Addr;
    MemInfo.size = Mem_Size;
    MemInfo.handle = 1;
    /* MemInfo.handle = 01; */
    __dpmi_physical_address_mapping(&MemInfo);

    /* Allocate LDT descriptor */
    Sel = __dpmi_allocate_ldt_descriptors(1);

```

```

/* Set Segment base addr & segment limit */
__dpmi_set_segment_base_address(Sel, MemInfo.address);
__dpmi_set_segment_limit(Sel, Mem_Size -1);
}

void Show_Select(void)
{
    /* clrscr(); */
    printf("\n\n1.Set WDT Time-out value and Enable WDT\n");
    printf("2.Retrigger WDT\n");
    printf("3.Disable WDT and Exit\n");
    printf("\n\nPress 1 - 3 :");
}

void reload_wdt(void)
{
    _farpokeb(Sel, Reload_Reg_offset, 0x80);
    _farpokeb(Sel, Reload_Reg_offset, 0x86);
    _farpokeb(Sel, Reload_Reg_offset+1, 0x01);
}

void disable_wdt(void)
{
    /* set WDT Lock Reg. */
    regs.x.ax = 0xb10b; /* set PCI Cfg. Byte */
    regs.x.di = 0x0068; /* get CR.0x68 */
    regs.h.cl = 0x00; /* WDT is WDT mode, WDT disable, Unlock */
    __dpmi_int(0x1a, &regs);
}

void enable_wdt(void)
{
    unsigned long i;
    /* Get PCI Device's Addr info */
    regs.x.ax = 0xb102; /* find PCI device */
    regs.x.dx = VendorID; /* Vender ID */
    regs.x.cx = DeviceID; /* Device ID */
    regs.x.si = i; /* Device index */
    __dpmi_int(0x1a, &regs);
    /* set WDT Configuration Reg. */
    regs.x.ax = 0xb10b; /* set PCI Cfg. Byte */
    regs.x.di = 0x0060; /* get CR.0x60 */
    regs.h.cl = 0x03; /* WDT_Out:enable, Clock is 1Khz, Disable INT when first stage reached */
    __dpmi_int(0x1a, &regs);

    printf("\n\nPress number 1 - 1000 to select time-out time(second): ");
    scanf("%d", &time_out);

    /* Preload Value 1 Reg. always set to about 1 Second */
    _farpokeb(Sel, Reload_Reg_offset, 0x80);
    _farpokeb(Sel, Reload_Reg_offset, 0x86);
    i = 0x000003e8;
    _farpokel(Sel, Preload_Value_1_offset, i);

    /* Preload Value 2 Reg. get by key press */
    _farpokeb(Sel, Reload_Reg_offset, 0x80);
    _farpokeb(Sel, Reload_Reg_offset, 0x86);
    i = (time_out * 1000) - 999;
    _farpokel(Sel, Preload_Value_2_offset, i);

    /* Do reload to make sure active */
    reload_wdt();

    /* set WDT Lock Reg. */

```



```

regs.x.ax = 0xb10b;    /* set PCI Cfg. Byte */
regs.x.di = 0x0068;    /* get CR.0x68 */
regs.h.cl = 0x02;      /* WDT is WDT mode, WDT enable, Unlock*/
__dpmi_int(0x1a, &regs);
}

int main(){
    int i,j;
    unsigned char k;

    Search_PCI_WDT();
    prepare_access_above_1mb();
    Show_Select();
    do
    {
        do
            i = getkey();
        while ((i <= KEY_1) && (i >= KEY_3));
        switch(i)
        {
            case KEY_1:
                enable_wdt();
                break;
            case KEY_2:
                reload_wdt();
                break;
            case KEY_3:
                disable_wdt();
                return(0);
                break;
        }
        Show_Select();
    }
    while(1);
}

```

## 2.19 GPIO Information

```

; For PPAP-3714 GPIO testing
;
; By Frank Hsu , 06/03/2004
;
; GPIO on PPAP-3714
;   PPAP-3714 J11_Pin1 =GPIO4:from HR_GPIO43
;   J11_Pin3 =GPIO3:from HR_GPIO42
;   J11_Pin5 =GPIO2:from HR_GPIO41
;   J11_Pin7 =GPIO1:from HR_GPIO40
;   J11_Pin9 =VCC
;   J11_Pin10=GND
;
;
; Programming Guide :
;
; PG_Step1 : Enable GPIO IO function and get GPIOBASE, then save to
;   ECX_Bit[31..16]
;
;   How to program GPIO[40,41,42,43]
;   -----
;   Get GPIOBASE =: B0:D31:F0:Offset[58..5Bh] ;(and let bit0 = 0 )
;   GPIO_CNTL =: B0:D31:F0:Offset_5Ch_bit4P1 ;Enable ICH4 GPIO
;
;   GPIO_USE_SEL2=:GPIOBASE + 30h, Bit[11..8]P[1,1,1,1]
;   GP_IO_SEL2  =:GPIOBASE + 34h, Bit[11..8],1-->Input,0-->Output
;

```

```
; GP_LVL2    =:(GPIOBASE + 38h, Bit[11..8],Input: readback,
;                                     output:1:high,0:low
;
; GPIO40 --> bit8,
; GPIO41 --> bit9,
; GPIO42 --> bit10,
; GPIO43 --> bit11,
;
; Let GPIO40,41,42,43 as GPIO pins by
; GPIO_USE_SEL2 (=(GPIOBASE + 30h))_bit[11..8]P[1,1,1,1];
;
; Define GPIO as input or output pin, 1=input pin, 0=output pin
; GP_IO_SEL2 (=(GPIOBASE + 34h))_bit[11..8]P[0/1,0/1,0/1,0/1];
;
; Output value or read status ?
; GP_LVL2    (=(GPIOBASE + 38h))_bit[11..8];
; Output value When output pin defined
; Read returned status when input pin defined
```

```
.MODEL tiny
.386
.STACK 200h
.data
```

PROMP1 DB'PORTWELL PPAP-3714 GPIO,3714GPIO.exe, V1.00 06-03-2004,All rights reserved.\$'

```
PROMP1_1 DB ' For PEB-3714 GPIO test ',13,10,'$'
PROMP_2_CR_LF db 0Dh, 0Ah,0Dh, 0Ah, '$'
PROMP_Str1 db ' PPAP-3714 J11_Pin1=GPIO4:from HR-ICH_GPIO43 ',0dh,0ah,'$'
PROMP_Str2 db ' J11_Pin3=GPIO3:from HR-ICH_GPIO42 ',0dh,0ah,'$'
PROMP_Str3 db ' J11_Pin5=GPIO2:from HR-ICH_GPIO41 ',0dh,0ah,'$'
PROMP_Str4 db ' J11_Pin7=GPIO1:from HR-ICH_GPIO40 ',0dh,0ah,'$'
PROMP_Str5 db ' ',0dh,0ah,'$'
PROMP_Str6 db ' Put a jumper cap on J11-pin1 and 3 , another jumper cap on pin5 and 7',0dh,0ah,'$'
PROMP_anykey db 0dh,0ah,' Ready ? If yes , then Press any key to start test ..... '$'

PROMP_err1 db 0dh,0ah,'"GPO43,41 write 1,1 to GPI42,40" failed.',0dh,0ah,'$'
PROMP_err2 db 0dh,0ah,'"GPO43,41 write 0,0 to GPI42,40" failed.',0dh,0ah,'$'
PROMP_err3 db 0dh,0ah,'"GPO42,40 write 1,1 to GPI43,41" failed.',0dh,0ah,'$'
PROMP_err4 db 0dh,0ah,'"GPO42,40 write 0,0 to GPI43,41" failed.',0dh,0ah,'$'
PROMP_TEST_OK db ' ^_^ ..... PPAP-3714 GPIO test ( 3714GPIO.EXE ) OK ..... ^_^',0dh,0ah,'$'
PROMP_TEST_fail db ' <<..... PPAP-3714 GPIO test ( 3714GPIO.EXE ) FAIL .....>>',0dh,0ah,'$'
PROMP_Qkey db 0dh,0ah,'Press "Q" key to stop test and return to DOS; or other key will go on next test.$'
```

```
GPIO_USE_SEL2_OFFSET db 30h ; The offset value from GPIOBASE
GP_IO_SEL2_OFFSET db 34h ; The offset value from GPIOBASE
GP_LVL2_OFFSET db 38h ; The offset value from GPIOBASE
; ECX_bit[31..16] save GPIOBASE ( B0:D31:F0:Offset[58..5Bh] )
```

```
.code
programstart:
```

```
mov ax,@data
mov ds,ax
```

```
lea dx,PROMP_2_CR_LF
mov ah,09h
int 21h
```

```
lea dx,PROMP1
mov ah,09h
int 21h
```

```
lea dx,PROMP_2_CR_LF
mov ah,09h
int 21h
```

```
lea dx,PROMP1_1
```

```

    mov ah,09h
    int 21h

    lea dx,PROMP_2_CR_LF
    mov ah,09h
    int 21h

    lea dx,PROMP_Str1
    mov ah,09h
    int 21h
    lea dx,PROMP_Str2
    mov ah,09h
    int 21h
    lea dx,PROMP_Str3
    mov ah,09h
    int 21h
    lea dx,PROMP_Str4
    mov ah,09h
    int 21h
    lea dx,PROMP_Str5
    mov ah,09h
    int 21h
    lea dx,PROMP_Str6
    mov ah,09h
    int 21h

    lea dx,PROMP_anykey ; ready to go ?
    mov ah,09h
    int 21h

    xor al,al
WAIT_KB_0:
    mov ah,1
    int 21h

    cmp al,0
    je WAIT_KB_0

    lea dx,PROMP_2_CR_LF
    mov ah,09h
    int 21h

    mov edx,00000000h ; Error flag in EDX_BIT[16..19], 0=ok, 1=failed

; PG_Step1 : Enable GPIO IO function and get GPIOBASE, then save to
;           ECX_Bit[31..16]
;
;
; ===== 1_start
; Get GPIOBASE Base Address , and save to ECX_bit[31..16]

    mov dx,0CF8h
    mov eax,8000F85Ch ; B0:D31:F0:Offset_5Ch
    out dx,eax
    mov dx,0CFCh
    in  eax,dx

    or  al,10h      ; 5Ch_Bit4P1 to Enable GPIO
    out dx,eax

    mov dx,0CF8h    ; Get GPIOBASE
    mov eax,8000F858h ; B0:D31:F0:Offset_58h

```

```

    out dx,eax
    mov dx,0CFCh
    in  eax,dx

    and al,0feh    ; bit 0 cleared to 0.
    rol  eax,10h
    mov ecx,eax    ; Save GPIOBASE to ECX[31..16]

; Get GPIOBASE Base Address , and save to ECX_bit[31..16]
; ===== 1_end

; Testing way :
; --- t1
; GPO43 Write 1 to GPI42 , GPI42 = 1 ? ,if yes, pass ; if no, failed
; GPO41 Write 1 to GPI40 , GPI40 = 1 ? ,if yes, pass ; if no, failed
; --- t2
; GPO43 Write 0 to GPI42 , GPI42 = 0 ? ,if yes, pass ; if no, failed
; GPO41 Write 0 to GPI40 , GPI40 = 0 ? ,if yes, pass ; if no, failed
;
; --- t3
; GPO42 Write 1 to GPI43 , GPI43 = 1 ? ,if yes, pass ; if no, failed
; GPO40 Write 1 to GPI41 , GPI41 = 1 ? ,if yes, pass ; if no, failed
; --- t4
; GPO42 Write 0 to GPI43 , GPI43 = 0 ? ,if yes, pass ; if no, failed
; GPO40 Write 0 to GPI41 , GPI41 = 0 ? ,if yes, pass ; if no, failed
;

    rol  ecx,10h ; Restore GPIOBASE from ECX[31..16] to ECX[15..0]

;
; Let GPIO43,42,41,40 as GPIO pins by
; GPIO_USE_SEL2 (=(GPIOBASE + 30h))_bit[11..8]P[1,1,1,1];

    xor bh,bh

    mov bl,GPIO_USE_SEL2_OFFSET ; Define as GPIO pins
    mov dx,cx                  ; GPIOBASE
    add dx,bx                   ; GPIO_USE_SEL2
    inc dx                      ;
    in  al,dx                   ; Read first
    or  al,0fh                  ; Set GPIO40~43 as GPIO pins
    out dx,al

; --- for t1 and t2 define GPO43,41 ; GPI42,40
; Define GPIO as input or output pin 1=input pin , 0=output pin
; GP_IO_SEL2 (=(GPIOBASE + 34h))_bit[11..8]P[0,1,0,1];

    mov bl,GP_IO_SEL2_OFFSET ; Define GPO,GPI pin
    mov dx,cx                  ; GPIOBASE
    add dx,bx                   ; GP_IO_SEL2
    inc dx                      ;
    in  al,dx                   ; Read First
    and al,0F0h                ; GPO43,41 , GPI42,40
    or  al,05h                  ;
    out dx,al

; ---- t1 start
; GPO43 Write 1 to GPI42 , GPI42 = 1 ? ,if yes, pass ; if no, failed
; GPO41 Write 1 to GPI40 , GPI40 = 1 ? ,if yes, pass ; if no, failed

;
; Output value or read status ?
; GP_LVL2 (=(GPIOBASE + 38h))_bit[3..0];
; Output value When output pin defined
; Read returned status when input pin defined

```

```

    mov bl,GP_LVL2_OFFSET    ; GPO43 Write 1 to GPI42
    mov dx,cx                ; GPO41 write 1 to GPI40
    add dx,bx                ; GP_LVL2
    inc dx                   ;
    in al,dx                 ; Read First
    or al,0Ah                ;
    out dx,al

    call IODELAY ; io delay

    in al,dx

    and al,05h ; Mask bit2,0 ( GPI42,40 )
    cmp al,05h
    je next_test1 ; okay , go on test

; no , error message display

    lea dx,prompt_err1
    mov ah,09h
    int 21h

    ror edx,10h ; error falg EDX_Bit16 , 1 --> Error happened
    or dl,01h
    rol edx,10h

    call KB_Wait

    next_test1 :

; ---- t1 end

; ---- t2 start
; GPO43 Write 0 to GPI42 , GPI42 = 0 ? ,if yes, pass ; if no, failed
; GPO41 Write 0 to GPI40 , GPI40 = 0 ? ,if yes, pass ; if no, failed

;      Output value or read status ?
;      GP_LVL2    (=(GPIOBASE + 38h))_bit[11..8];
;      Output value When output pin defined
;      Read returned status when input pin defined

    mov bl,GP_LVL2_OFFSET    ; GPO43 Write 0 to GPI42
    mov dx,cx                ; GPO41 write 0 to GPI40
    add dx,bx                ; GP_LVL2
    inc dx                   ;
    in al,dx                 ; Read First
    and al,0F5h              ;

    call IODELAY ; io delay

    out dx,al

    call IODELAY ; io delay

    in al,dx

    and al,05h ; Mask bit 2,0
    cmp al,00h
    je next_test2 ; okay , go on test

; no , error message display

```

```

    lea dx,prompt_err2
    mov ah,09h
    int 21h

    ror edx,10h ; error falg EDX_Bit17 , 1 --> Error happened
    or dl,02h
    rol edx,10h

    call KB_Wait

    next_test2 :

; ---- t2 end

; --- for t3 and t4 define GPI43,41 ; GPO42,40
; Define GPIO as input or output pin 1=input pin , 0=output pin
; GP_IO_SEL2 (=(GPIOBASE + 34h))_bit[11..8]P[1,0,1,0];

    mov bl,GP_IO_SEL2_OFFSET ; Define GPO,GPI pin
    mov dx,cx
    add dx,bx ; GP_IO_SEL2
    inc dx ;
    in al,dx ; Read First
    and al,0F0h ; GPI43,41 , GPO42,40
    or al,0Ah ;

    call IODELAY ; io delay

    out dx,al

; ---- t3 start
; GPO42 Write 1 to GPI43 , GPI43 = 1 ? ,if yes, pass ; if no, failed
; GPO40 Write 1 to GPI41 , GPI41 = 1 ? ,if yes, pass ; if no, failed

; Output value or read status ?
; GP_LVL2 (=(GPIOBASE + 38h))_bit[11..8];
; Output value When output pin defined
; Read returned status when input pin defined

    mov bl,GP_LVL2_OFFSET ; GPO42 Write 1 to GPI43
    mov dx,cx ; GPO40 write 1 to GPI41
    add dx,bx ; GP_LVL2
    inc dx ;
    in al,dx ; Read First
    or al,05h ;

    call IODELAY ; io delay

    out dx,al

    call IODELAY ; io delay

    in al,dx

    and al,0ah ; Mask bit3,1 GPI43,41
    cmp al,0ah
    je next_test3 ; okay , go on test

; no , error message display

    lea dx,prompt_err3
    mov ah,09h
    int 21h

```

```

        ror edx,10h ; error falg EDX_Bit18 , 1 --> Error happened
        or  dl,04h
        rol  edx,10h

        call KB_Wait

        next_test3 :

; ---- t3 end

; ---- t4 start
; GPO42 Write 0 to GPI43 , GPI43 = 0 ? ,if yes, pass ; if no, failed
; GPO40 Write 0 to GPI41 , GPI41 = 0 ? ,if yes, pass ; if no, failed

;      Output value or read status ?
;      GP_LVL2      (=(GPIOBASE + 38h))_bit[11..8];
;      Output value When output pin defined
;      Read returned status when input pin defined

        mov bl,GP_LVL2_OFFSET ; GPO42 Write 0 to GPI43
        mov dx,cx ; GPO40 write 0 to GPI41
        add dx,bx ; GP_LVL2
        inc dx ;
        in  al,dx ; Read First
        and al,0FAh ;

        call IODELAY ; io delay

        out dx,al

        call IODELAY ; io delay

        in  al,dx

        and al,0ah ; Mask bit 3,1
        cmp al,00h
        je test_end ; okay , end of test

; no , error message display

        lea dx,prompt_err4
        mov ah,09h
        int 21h

        ror edx,10h ; error falg EDX_Bit19 , 1 --> Error happened
        or  dl,08h
        rol  edx,10h

; ---- t4 end

test_end :

        ror edx,10h
        cmp dl,00h
        je test_ok

test_fail :

        lea dx,PROMP_2_CR_LF
        mov ah,09h
        int 21h

        lea dx,prompt_TEST_fail

```

```

        mov ah,09h
        int 21h

        jmp return_to_dos

test_ok :
        lea dx,prompt_TEST_OK
        mov ah,09h
        int 21h

        ror ecx,10h          ; ECX[15..0] to ECX[31..16]
                                ; Restore GPIOBASE to ECX[31..16]
return_to_dos :
        mov ah,4ch ; Return to DOS
        int 21h

IODELAY PROC    near
        push ax
        push dx

        mov dx,0edh
        in  al,dx
        jmp $+2
        mov dx,0edh
        in  al,dx

        pop dx
        pop ax
        ret
IODELAY ENDP

KB_wait PROC    near

        push ax
        push bx
        push cx
        push dx

        lea dx,PROMPT_Qkey
        mov ah,9             ; Display "Q" key prompt
        int 21h

        xor al,al
WAIT_KB:
        mov ah,1
        int 21h

        cmp al,0
        je  WAIT_KB

        cmp al,51h ; "Q" pressed ?
        je  test_fail
        cmp al,71h ; "q" pressed ?
        jne call_return
        jmp test_fail ;

call_return :

        pop dx
        pop cx
        pop bx
        pop ax

        ret

```



KB\_wait ENDP

END programstart

## *2-20 About EZIO-100*

Proprietary keypad and LCD display interfaces are implemented in traditional computing system design, but they are usually different from system to system. The main purpose to roll this module out is to provide an easier human-machine interface for those computing systems regarding application friendly operation as a “must.”

The design goals of this interface are:

- ◆ A single interface for those applications where both LCD display and keypad are required.
- ◆ This interface should be available in every computing system.
- ◆ The communication implementation should be OS independent.

Our solution is to use “Serial port” as the interface for both LCD display and keypad. A simple protocol is further defined so that applications can directly communicate with this module no matter what the Operating System is.

### **WARNING!**

**THE LCD DRIVER ICS ARE MADE OF CMOS PROCESS, DAMAGED BY STATIC CHARGE VERY EASILY. MAKE SURE THE USER IS GROUNDED WHEN HANDLING THE LCD.**

## *2-20.1 Features*

- Ideal user interface for communication appliance
- No driver required; OS independent
- Alphanumeric characters display support
- Four key pads can be customized for different applications
- Easy system installation and operation
- Clearly display system status
- Single interface to SBC or M/B

## *2-20.2 Technical Support Information*

For further support, users may also contact Portwell’s headquarter in Taipei or your local distributors.

***Taipei Office Phone Number: +886-2-27992020***

## 2-20.3 Mechanical Specification

<b>Module Size (mm):</b>	• 101.6(W) x 26.0(H) x 30.6(D) (max.)
<b>Display Format:</b>	• 16 characters x 2 lines
<b>Character Size:</b>	• 3.0 x 5.23 mm

## 2-20.4 General Specification



### General Specification

<b>Display Resolution:</b>	• 16 characters x 2 lines
<b>Dimensional Outline (mm):</b>	• 101.6(W) x 26.0(H) x 30.6(D) (max.)
<b>Function Key:</b>	• Four operation keys (up, down, enter and ESC)
<b>Display Icon:</b>	• Eight self-defined icons
<b>Interface:</b>	• RS-232



### Absolute Maximum Rating

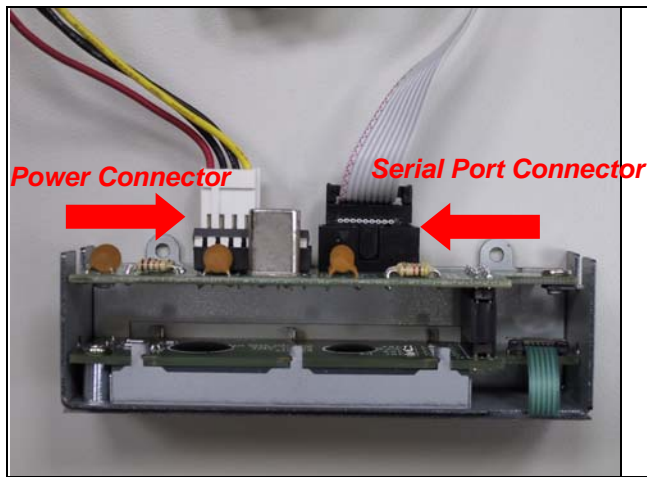
Item	Normal Temperature			
	Operating		Storage	
	Max.	Min.	Max.	Min.
<b>Ambient Temperature</b>	0°C	+50°C	-20°C	+70°C
<b>Humidity (w/o condensation)</b>	Note 2, 4		Note 3, 5	

## 2-20.5 Product Outlook



## 2-20.6 Interface Pin Assignment

There are only two connectors in this module, as shown in **Figure C.2-1**: power connector and Serial Port connector. The power source into this module is 5 volt only. There are only three pins used in the Serial Port interface (**Figure C.2-2**).



**Fig. C.2-1** Power connector and serial port connector of EZIO-100

5	4	3	2	1
10	9	8	7	6

Pin 2: TxD Pin 3 : RxD Pin 5 : Ground

**Fig. C.2-2** Pin assignment

In other words, the Serial Port is defined as DCE. Therefore, we can use a straight-through cable to connect it to the Serial Port of most of the computers, defined as DTE.

#### (1) Interface Pin Assignment

PIN NO.	PIN OUT	Description
1	NC	No connector
2	RXD	RS232 Data
3	TXD	RS232 Data
4	NC	No connector
5	V <sub>SS</sub>	Ground
6	NC	No connector
7	NC	No connector
8	NC	No connector
9	NC	No connector
9	NC	No connector

#### (2) Power

PIN NO.	PIN OUT	Description
1	NC	No connector
2	GND	Power GND
3	GND	Power GND
4	+5V	Power VCC (+5V)

## 2-20.7 Hardware installation

The installation steps are:

- ◆ Connect the power connector to the power connector of this module.
- ◆ Connect the straight-through cable between Serial Port of this module and computer

First, all versions (00A, 01A, 02A) of EZIO can use those commands. Only the 02A version of EZIO firmware that adds “FE 28” & “FE 37” command can control start of HEX & End of HEX.

EZIO is an intelligent device, which will display those data received from RS-232 port and reply key pressing status to polling command from RS-232 port. Both commands and data go thru RS-232 ports. To distinguish between data and commands, the LCD/key-pad module recognizes a command prefix, 254 (Hex 0FE). The byte following “254” will be processed as a command. For example, to clear the screen, send the command prefix (254) followed by the LCD clear-screen code (1). The valid data range is shown as the following table:

<i>Valid data range</i>	<i>Displayed characters</i>
0-7	Customized icon 0-7
48-57 (30-39 Hex)	0-9
65-90 (41-5A Hex)	A-Z
97-122 (61-7A Hex)	a-z

To get the key pressing status, a “read key” command can be issued to this module, which will check the key-pressing status and reply accordingly. The following are the commands and corresponding Decimal/Hex values:

	<i>Functions/commands</i>	<i>Decimal/Hex</i>	<i>Comment</i>
1.	Start Of HEX	40/28	Only for 02A
2.	End Of HEX	55/37	Only for 02A
3.	Clear screen	1/01	
4.	Home cursor	2/02	
5.	Read key	6/06	See note 1
6.	Blank display (retaining data)	8/08	
7.	Hide cursor & display blanked characters	12/0C	
8.	Turn on (blinking block cursor)	13/0D	
9.	Show underline cursor	14/0E	
10.	Move cursor 1 character left	16/10	
11.	Move cursor 1 character right	20/14	
12.	Scroll 1 character left	24/18	
13.	Scroll 1 character right	28/1C	
14.	Set display address (position the cursor) location	128 (Hex080)+ Location	See note 2
15.	Set character-generator address	64 (Hex 040)+ address	See note 3

*Note 1: Upon receiving the “read key” command from host computer, the LCD/keypad module will check the status of every key and reply with status command accordingly. The replied message from LCD/key-pad module consists of a header and a status byte. The header byte is 253 (Hex0FD). The high nibble (with the most significant bit) of the status byte is always “4” and the low nibble (with the least significant bit) of the status byte is used to indicate key pressing status of the keypad module. This nibble will be “F” (of four 1s), if no key pressed while the “read key” received. “0” will be used to indicate key pressing status of corresponding key. There are four keys in this module – upper arrow, down arrow, enter (ENT), and escape (ESC). The relationship between the function key, corresponding status bit and status byte is shown as the table below.*

Function key	Corresponding status bit	Status byte
Escape	The fourth bit of lower nibble (the least significant bit) (1110)	4E (H)
Up arrow	The third bit of lower nibble (1101)	4D (H)
Enter	The second bit of lower nibble (1011)	4B (H)
Down arrow	The first bit of lower nibble (0111)	47 (H)

More than one key can be pressed at the same time so that there may be more than one "0"s in the low nibble of status byte. For example, if Up and Down arrow keys are pressed at the same time while "read key" command received, the replied status will be "Hex045".

Note 2: This command can be used to place the cursor at any location. The corresponding address for each character on the screen is as follows:

#### For 16x2 Display Address

Character	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Location	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
(Address)	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F

The addresses of characters at the same row are continuous, so moving cursor commands can be applied to shift the cursor position back and forth. However, the addresses of characters between upper and lower row are discontinuous. To change cursor position between upper row and lower row, this command will be applied.

Note 3: This command can be used to create customized icon. The starting address is 64 and every character will take 8 bytes to create a 5(W) x 7(H) resolution picture, as shown below:

#### CG RAM MAPPING

CG RAM Address						Character Patterns (CG RAM data)									
5	4	3	2	1	0	7	6	5	4	3	2	1	0		
High			Low			High			Low						
0	0	0	0	0	0	*	*	*	0	1	1	0	0	← Character Pattern	
			0	0	1				1	0	0	1	0		
			0	1	0				0	0	1	0	0		
			0	1	1				0	1	0	0	0		
			1	0	0				1	1	1	1	0		
			1	0	1				0	0	0	0	0		
			1	1	0				0	0	0	0	0		
			1	1	1				0	0	0	0	0		
0	0	1	0	0	0	*	*	*	1	1	1	1	1	← Character Pattern	
			0	0	1				1	0	0	0	1		
			0	1	0				1	0	1	0	1		
			0	1	1				1	0	1	1	1		
			1	0	0				1	0	1	0	1		
			1	0	1				1	0	0	0	1		
			1	1	0				1	1	1	1	1		
			1	1	1				0	0	0	0	0		
1	1	1	0	0	0	*	*	*	1	1	1	1	1	← Character Pattern	
			0	0	1				1	0	0	0	1		
			0	1	0				1	1	1	0	1		
			0	1	1				1	0	0	0	1		
			1	0	0				1	0	1	1	1		
			1	0	1				1	0	0	0	1		
			1	1	0				1	1	1	1	1		
			1	1	1				1	1	1	1	1		



To show the customized icon, simply send the data between “0” to “7” to this module.

For example, this module will display the customized icon at location 64 to 71 upon receiving data “0”; while it will display the customized icon at location 72 to 79 upon receiving data “1”.

Watchdog timer is also built in the module. This module will reset itself and send out “reset packet” (0FDH, 0EH) thereafter.

The input must be a standard RS-232 or inverted TTL signal. The RS-232 setting should be:

- ♦ Baud rate: 2400 bps
- ♦ Parity: None
- ♦ Data bits: 8
- ♦ Stop bit: 1

What follows is the default setup after LCD module initiated:

- ♦ 2-line display mode; every character is 5 x 8 dots.
- ♦ Display on; cursor off; cursor blink off.
- ♦ Display will be cleared.
- ♦ Shift right for entry mode.
- ♦ Set address counter to “00”(cursor position to 0)
- ♦ In entry mode.

## 2-20.9 Character Generator ROM (CGROM)

Upper bits Lower bits		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)			0	a	P	`	P					—	3	E	a	p
0001	CG RAM (2)		!	1	A	D	a	4				o	7	+	4	a	q
0010	CG RAM (3)		"	2	B	R	b	r				r	/	"	x	p	e
0011	CG RAM (4)		#	3	C	S	c	s				J	U	T	E	s	e
0100	CG RAM (5)		\$	4	D	T	d	t				\	I	T	+	p	a
0101	CG RAM (6)		%	5	E	U	e	u				•	o	+	1	e	o
0110	CG RAM (7)		&	6	F	V	f	v				9	o	+	3	p	z
0111	CG RAM (8)		'	7	G	W	g	w				7	+	7	9	g	w
1000	CG RAM (1)		(	8	H	X	h	x				4	o	+	1	7	x
1001	CG RAM (2)		)	9	I	Y	i	y				e	T	1	u	7	y
1010	CG RAM (3)		*	:	J	Z	j	z				x	o	+	1	7	z
1011	CG RAM (4)		+	;	K	[	k	[				7	+	1	o	7	z
1100	CG RAM (5)		,	<	L	]	l	]				7	+	1	o	7	z
1101	CG RAM (6)		—	=	M	^	m	^				7	+	1	o	7	z
1110	CG RAM (7)		.	>	N	_	n	_				7	+	1	o	7	z
1111	CG RAM (8)		/	?	O	__	o	+				7	+	1	o	7	z

## 2-20.10 Sample Codes

```
/* *****
* EZIO RS232 LCD Control Sample Program
* *****
* *****

* Company:      Portwell Inc.
* Date:         4/16/2003
* Program:      02A.c
* Version:      1.02
* Compile:      Linux GNU C
* Purpose:      Direct access to EZIO LCD, the program will display
*               messages according to the control button. The current
*               version only has the following function:
*
*               1: display welcome message
*               2: display UP message if "scroll up" button is pressed
*               3: display ENTER message if "ENTER" button is pressed
*               4: display ESC message if "ESC" button is pressed
*               5: display DOWN message if "scroll down" button is pressed
*
* Program Overview:
*
*   - Parameters:
*       fd          : a file name for open() method, here represents the com port
*       Cmd         : command prefix
*       cls         : clear command
*       init        : initialize command
*       blank       : display blank screen
*       stopsend    : stop input/output
*       home        : move cursor to initial position
*       readkey: set to read from EZIO
*       hide        : hide cursor & display blanked characters
*       movel       : move cursor one character left
*       mover       : move cursor one character right
*       turn        : turn on blinking-block cursor
*       show        : turn on underline cursor
*       scl         : scroll cursor one character left
*       scr         : scroll cursor one character right
*       setdis      : set character-generator address
*
*   - Procedure:
*       1. The program sets up the environment, i.e. com port settings.
*       2. The main function MUST call init() twice to initialize EZIO
*          before any communication.
*       3. For executing any command, the command prefix, Cmd, MUST be
*          called be command. So all command contains two parts, eg.
*          to initialize the sequence of HEX number is 0xFE, 0x25.
*       4. After clear screen and display welcome message, ReadKey()
*          method must be call to advise EZIO for reading data.
*       5. A pooling method is implemented to get input from EZIO while
*          any button is pressed.
*
*   - NOTE: This program is a sample program provided " AS IS" with NO
*           warranty.
*
* Copyright (c) Portwell, Inc. All Rights Reserved.
* *****/
```



```

#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <stdlib.h>

static int fd;

void SetEnvironment () {
    system("stty ispeed 2400 < /dev/ttyS1");
    system("stty raw < /dev/ttyS1");
}

int Cmd = 254; /* EZIO Command */
int cls = 1; /* Clear screen */
void Cls () {
    write(fd,&Cmd,1);
    write(fd,&cls,1);
}

int init = 0x28;
void Init () {
    write(fd,&Cmd,1);
    write(fd,&init,1);
}

int stopsend = 0x37;
void StopSend () {
    write(fd,&Cmd,1);
    write(fd,&init,1);
}

int home = 2 ; /* Home cursor */
void Home () {
    write(fd,&Cmd,1);
    write(fd,&home,1);
}

int readkey = 6 ; /* Read key */
void ReadKey () {
    write(fd,&Cmd,1);
    write(fd,&readkey,1);
}

int blank = 8 ; /* Blank display */
void Blank () {
    write(fd,&Cmd,1);
    write(fd,&blank,1);
}

int hide = 12 ; /* Hide cursor & display blanked characters */
void Hide () {
    write(fd,&Cmd,1);
    write(fd,&hide,1);
}

int turn = 13 ; /* Turn On (blinking block cursor) */
void TurnOn () {
    write(fd,&Cmd,1);
    write(fd,&turn,1);
}

int show = 14 ; /* Show underline cursor */
void Show () {
    write(fd,&Cmd,1);
    write(fd,&show,1);
}

```

```

}

int moveL = 16 ; /* Move cursor 1 character left */
void MoveL () {
    write(fd,&Cmd,1);
    write(fd,&moveL,1);
}

int mover = 20 ; /* Move cursor 1 character right */
void MoveR () {
    write(fd,&Cmd,1);
    write(fd,&mover,1);
}

int scl = 24;      /* Scroll cursor 1 character left */
void ScrollL(){
    write(fd,&Cmd,1);
    write(fd,&scl,1);
}

int scr = 28;      /* Scroll cursor 1 character right */
void ScrollR(){
    write(fd,&Cmd,1);
    write(fd,&scr,1);
}

int setdis = 64; /* Command */
void SetDis(){
    write(fd,&Cmd,1);
    write(fd,&setdis,1);
}

}

/* Add or Change Show Message here */
char mes1[] = "Portwell EZIO";
char mes2[] = "*****";
char mes3[] = "Up is selected";
char mes4[] = "Down is selected";
char mes5[] = "Enter is selected";
char mes6[] = "ESC is selected";
char nul[] = "          ";

int a,b;
void ShowMessage (char *str1 , char *str2) {
    a = strlen(str1);
    b = 40 - a;
    write(fd,str1,a);
    write(fd,nul,b);
    write(fd,str2,strlen(str2));
}

int main () {

    SetEnvironment(); /* Set RAW mode */

    fd = open("/dev/ttyS1" ,O_RDWR); /** Open Serial port (COM2) */

    Init(); /* Initialize EZIO twice */
    Init();

    Cls(); /* Clear screen */
    ShowMessage(mes1,mes2);

```

```

while (1) {
    int res;
    char buf[255];

    SetDis();
    ReadKey(); /* sub-routine to send "read key" command */
    res = read(fd,buf,255); /* read response from EZIO */

    switch(buf[1]) {      /* Switch the Read command */

        case 0x4D :      /* Up Botton was received */
            Cls();
            ShowMessage(mes1,mes3); /* display "Portwell EZIO" */
            break;        /* display "Up is selected" */

        case 0x47 :      /* Down Botton was received */
            Cls();
            ShowMessage(mes1,mes4); /* display "Portwell EZIO" */
            break;        /* display "Down is selected" */

        case 0x4B :      /* Enter Botton was received */
            Cls();
            ShowMessage(mes1,mes5); /* display "Portwell EZIO" */
            break;        /* display "Enter is selected" */

        case 0x4E :      /* Escape Botton was received */
            Cls();
            ShowMessage(mes1,mes6); /* display "Portwell EZIO" */
            break;        /* display "Escape is selected" */

    }
}
}

```

## Chapter 3      Operation Guide

### 3.1    *Brief Guide of PPAP-3714L*

PPAP-3714L is an Intel®Pentium™ 4 based board with Cavium Nitrox Lite CN1010X supporting IPSec/SSL. The onboard Cavium security processor provides a cost advantage and the flexibility to implement various networking appliances. The PPAP-3714L has the capability of processing up 1GB IPSec traffic or 7000 RSA operations per second.

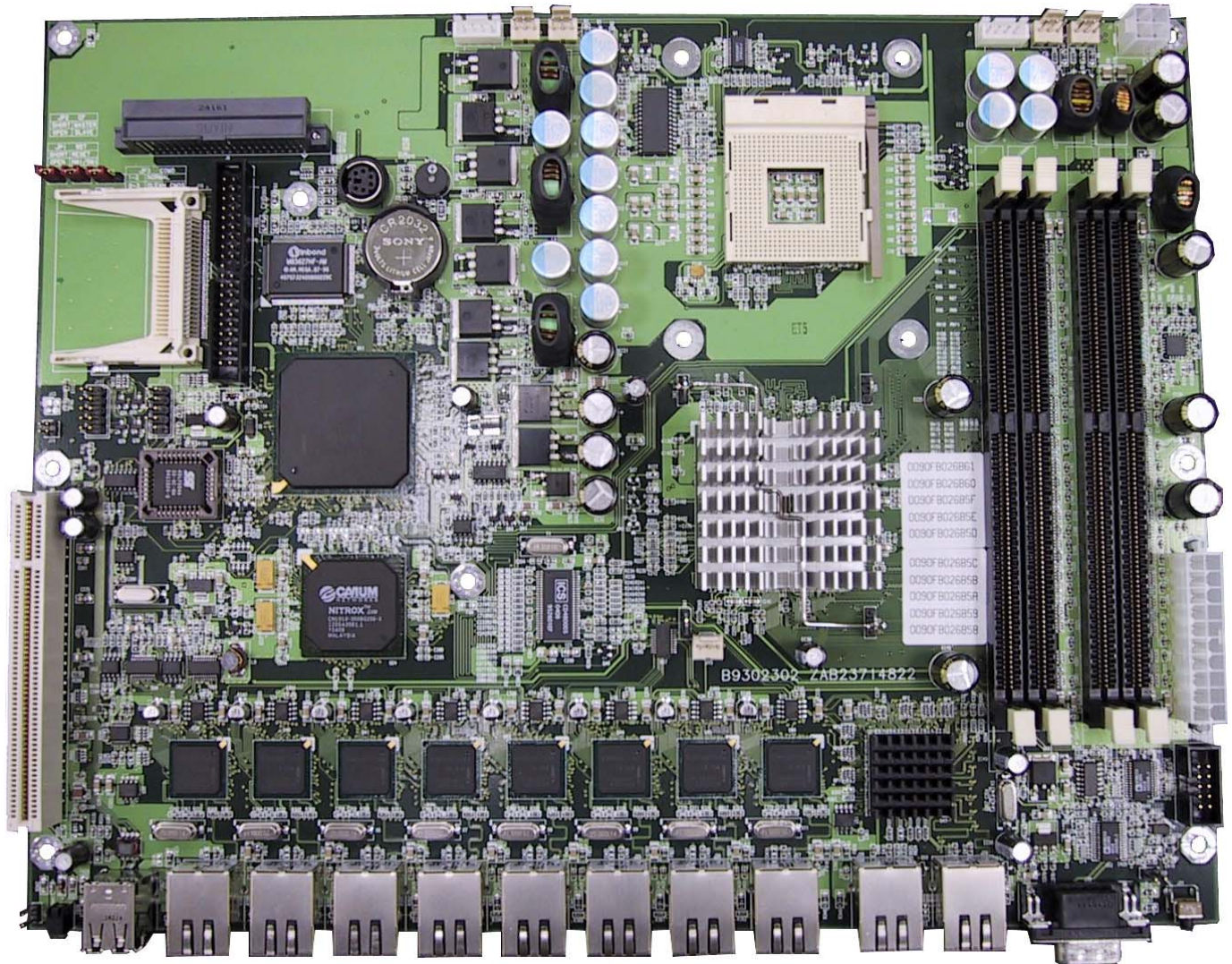
PPAP-3714L is a Communication Appliance computing board based on Intel 875P chipset technology. PPAP-3714L has ten on-board LAN ports to serve communication appliances, such as Firewall, which needs sufficient ethernet ports to connect external network (internet), demilitarized zone and internal network. Different I/O management policies can be applied respectively to individual network to achieve the highest security level. The target market segment is communication appliance including Virtual Private Network, Load Balancing, Quality of Service, Intrusion Detection, Virus Detection, Firewall and Voice Over IP.

This PPAP-3714L system board is utilized with Intel® mPGA Celeron® and Intel® mPGA Pentium® 4 processors, and 184-pin DIMM. The enhanced on-board PCI IDE interface supports 2 drives up to PIO mode 4 timing and Ultra DMA/100 synchronous mode feature. The on-board

Intel 6300ESB chipset integrates two serial ports driven by two high performance 16C550-compatible UARTs to provide 16-byte send/receive FIFOs. In addition, the two Universal Serial Bus ports provide high-speed data communication between peripherals and PC.

The on-board flash ROM is used to make the BIOS update easier. The high precision Real Time Clock/Calendar is built to support Y2K for accurate scheduling and storing configuration information. All of these features make PPAP-3714L excellent in stand-alone applications.

If any of these items is damaged or missing, please contact your vendor and save all packing materials for future replacement and maintenance.



**Figure 3-1 PPAP-3714L Board**

## 3.2 System Architecture

The following illustration of block diagram illustrated basic design reference of PPAP-3714L, a highly integrated system solution. The most up-to-date system architecture of PPAP-3714L includes two main chips. It contains and to support mPGA2 Pentium 4 processor, DIMM, PCI bus interface, USB port, SMBus communication, and Ultra DMA/100 IDE Master.

PPAP-3714L has built-in Socket 478 to support Intel mPGA Celeron/Pentium 4 processor (533 / 800MHz FSB) for cost-effective and high performance application.

The provides a completely integrated solution for the system controller and data path components in a Celeron/Pentium 4 processor system. It provides optimized 64-bit DDRAM

interface with one 184 pin 2.5V DIMM.

The provides a highly integrated multifunction for the best industry applications. It supports 2-channel dedicated Ultra ATA/33/66/100 IDE master interface, Universal Serial Bus (**USB**) controllers and two 32-bit PCI bus interfaces.

All detailed operating relations are shown in **Fig. 3-2** (PPAP-3714L System Block Diagram).

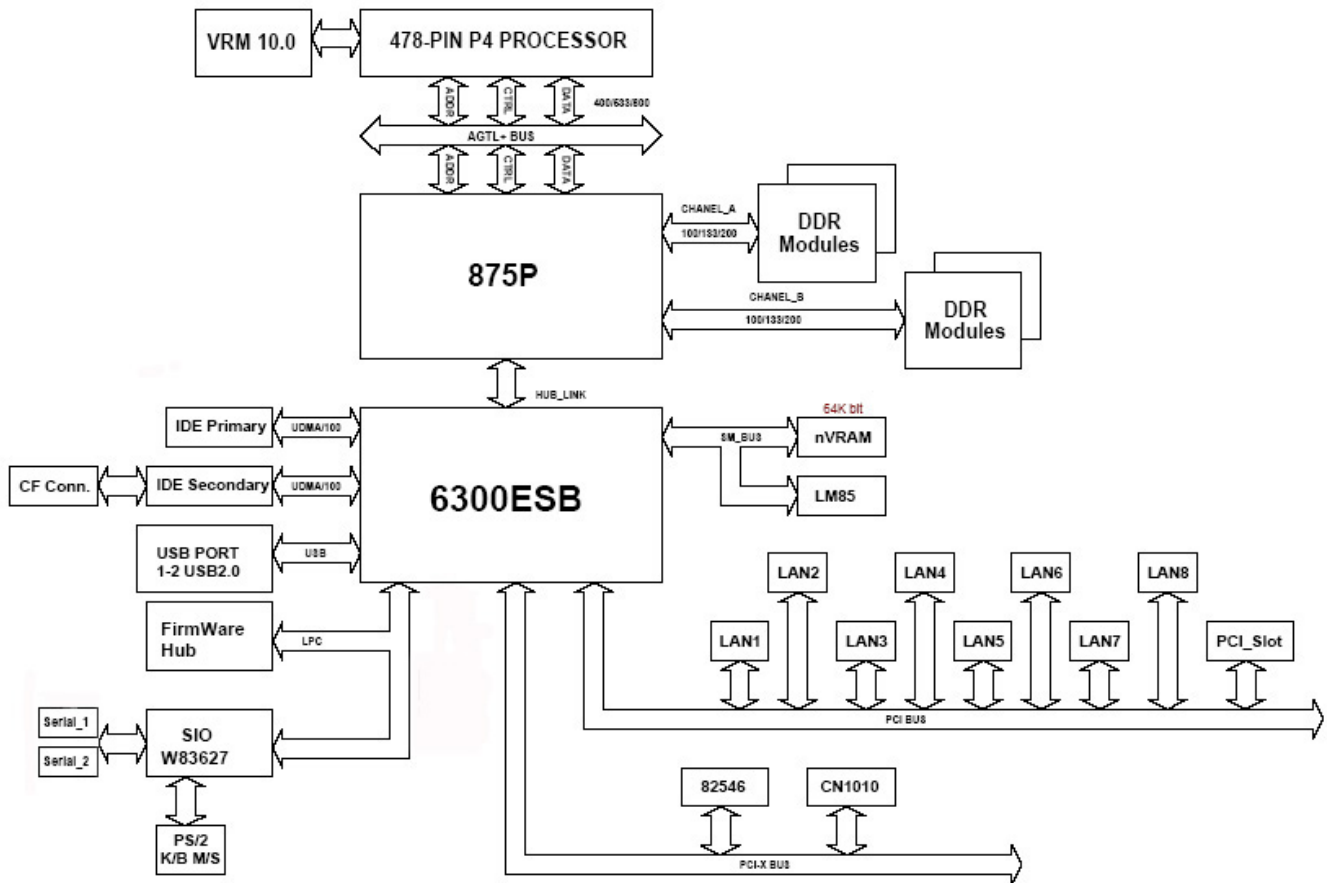


Figure 3-2 PPAP-3714L Block Diagram